Neuronale Identifikation von Totzeiten

Claudia Brand

Institut für Informatik
der Technischen Universität München

Neuronale Identifikation von Totzeiten

Claudia Brand

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende:	Univ.-Prof. G. J. Klinker, Ph.D.
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. Dr. h.c. W. Brauer
2.	Univ.-Prof. (komm.) Dr. E. Jessen, em.

Die Dissertation wurde am 12.3.2002 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 14.6.2002 angenommen.

Pro Babi

Kurzfassung

Der Umgang mit Totzeiten bzw. Totzeitsystemen stellt in der Regelungstechnik ein schwieriges Problem dar. Ein System ist mit Totzeiten behaftet, wenn es mindestens eine Komponente enthält, deren Eingaben nicht sofort, sondern erst nach einer bestimmten Zeitverzögerung die Ausgaben beeinflussen. In realen Systemen kommen derartige Totzeiten zahlreich vor und sie erschweren die Systemregelung zum Teil erheblich.

Der Totzeitproblematik wird bisher ungenügend Rechnung getragen. Bei der Modellierung eines Systems werden Totzeiten häufig einfach ignoriert, da ihre Größe unbekannt und ihre Identifikation sehr problematisch ist. Dies gilt insbesondere für nichtlineare dynamische Systeme. Als Konsequenz daraus ergibt sich zwangsläufig eine suboptimale Regelung des Systems.

Die vorliegende Arbeit leistet einen Beitrag zur Lösung der Totzeitproblematik. Es wird eine neue Methode zur Identifikation von beliebigen Totzeiten in nichtlinearen dynamischen Systemen und zur verbesserten Modellierung und Regelung von Totzeitsystemen vorgestellt. Diese Methode verwendet zum einen speziell für diese Zwecke entwickelte neuronale Netze und zum anderen spezielle Konfidenzmaße zur Beurteilung der Zuverläßigkeit der Identifikation. Die Methode erlaubt eine einfache Integration von eventuell vorhandenem Vorwissen über die zu identifizierenden Totzeiten. Der erfolgreiche Einsatz dieser neuen Methode wird durch Anwendungsbeispiele aus der Mikroelektronik, der Automobilindustrie und der Lebensmitteltechnologie gezeigt.

Abstract

The research presented in this work addresses the time-delay problem in control systems. A system is considered to be a time-delay system, if it contains at least one component, whose input affects the output not immediately but after a certain time delay. Such time-delay components often appear in control systems and have a negative influence on control.

Presently, time delays are often ignored in the design of control systems since they are unknown and their identification is difficult. This holds especially for nonlinear dynamic systems. As a consequence, the system cannot be optimally controlled.

This work contributes to solve the time-delay problem. A new method for the identification of arbitrary time delays in nonlinear dynamic systems is presented. This method uses neural networks, that are especially designed for this task, together with particular criteria for the evaluation of the identification reliability. The method enables an easy integration of a priori knowlegde about the time delays. Experimental results are presented showing the successful application of this new method in the fields of microelectronics, automotive industry and food technology.

Danksagung

Mein besonders herzlicher Dank gilt an erster Stelle meinem Doktorvater Herrn Professor Dr. Dr. h.c. Wilfried Brauer, für die stete Förderung und Unterstützung meiner Arbeit. Er gab mir wertvolle Hinweise und Anregungen und gewährte mir zugleich den nötigen Freiraum zur Verwirklichung meiner eigenen Ideen.

Herrn Professor Dr. Eike Jessen möchte ich sehr herzlich dafür danken, daß er sich als Zweitgutachter zur Verfügung gestellt hat.

Eine optimale Arbeitsatmosphäre bot mir die Forschungsgruppe KI/Kognition am Lehrstuhl Prof. Brauer. Besonders bedanke ich mich für die Zusammenarbeit bei Michael Sturm. Stets war er mit Rat und Tat zur Stelle, in allen möglichen Belangen — eine optimalere Zusammenarbeit kann ich mir nicht vorstellen. Ein ebenso herzlicher Dank richtet sich an meinen Zimmerkollegen Clemens Kirchmair für seine fachliche Unterstützung und vor allem für den optimalen Spaßfaktor im Büro! Meinem „Motivator“ Stefan Kriebel danke ich ganz besonders für seine Unterstützung in zahlreichen fachlichen, organisatorischen und persönlichen Belangen. Ich danke Michael, Clemens und Stefan außerdem sehr herzlich für die mühevolle Arbeit des Korrekturlesens und ihre wertvollen Hinweise.

Weiterhin möchte ich mich für die kollegiale Zusammenarbeit bedanken bei Volker Baier, Felix Brandt, Matthias Nickles, Michael Rovatsos, Klaus Stein und Gerhard Weiß, sowie den ehemaligen KI'lern Dieter Bartmann, Till Brychcy, Sepp Hochreiter und Alexandra Musto. Iris Gilsdorf möchte ich für ihre zahlreichen Ratschläge danken.

Eine Promotion erfordert nicht nur fachlichen, sondern auch viel privaten Rückhalt. Es gibt eine Reihe von Freunden und Bekannten, die für den absolut notwendigen Ausgleich zur Arbeit gesorgt haben und mich auf vielfältige Art und Weise unterstützten und motivierten. Ihnen allen gilt mein besonders herzlicher Dank.

In diesem Zusammenhang danke ich in ganz besonderer Weise meiner Familie, insbesondere meinen Eltern und meiner Schwester, für die viele Kraft und Geborgenheit.

Abschließend möchte ich mich bei meinem Mann Matthias bedanken, für all seine liebevolle Unterstützung und seine unendliche Geduld.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Das Streben nach Präzision ist ein ureigenes Merkmal der technisierten Gesellschaft. Dem stehen jedoch die heute immer noch stark beschränkten Möglichkeiten zur Modellierung technischer Systeme gegenüber. War es bis vor kurzem beispielsweise vertretbar bzw. mangels ausreichender Rechenkapazität gar nicht anders möglich, von annähernd linearen Systemen als Modellierungsgrundlage auszugehen, so verlangen Qualitätssteigerungen auf dem derzeitigen Stand der Technik eine viel genauere Betrachtungsweise, z.B. die Berücksichtigung auch der Nichtlinearitäten in den zu modellierenden Systemen.

Die Kraftfahrzeug- und insbesondere Motorentechnik etwa wird zunehmend vom gestiegenen Umweltbewußtsein beeinflußt. Dies äußert sich in immer schärferen Richtlinien, beispielsweise in strengen Grenzwerten für Abgasemissionen. Folglich geht die Entwicklung hin zu verbrauchsoptimierten und die Umwelt weniger belastenden Arbeitsprozessen und Produkten — eine Entwicklung, die nicht allein auf die Automobilindustrie beschränkt ist. Das Thema Optimierung zieht sich in den unterschiedlichsten Ausprägungen durch alle Industriebranchen. Unterstützt wird dieser Prozeß in den letzten Jahren von der Entwicklung von systemtechnischen Verfahren, deren Ziel die Verbesserung von Produktions- und Betriebsbedingungen ist. Klassische Verfahren können mit dieser Entwicklung kaum Schritt halten. Neue Wege durch den Einsatz intelligenter Verfahren, wie in dieser Arbeit verfolgt, sind in diesem Zusammenhang richtungsweisend.

Diese Arbeit ist interdisziplinär und steht an der Schnittstelle zweier wissenschaftlicher Bereiche, nämlich Neuroinformatik und Regelungstechnik. Die Neuroinformatik befaßt sich mit der Modellierung von linearen und nichtlinearen Systemen durch eine konzeptionelle Simulation von natürlichen neuronalen Netzen. Wo konventionelle Methoden zu aufwendig oder nicht mehr anwendbar werden, erreichen künstliche neuronale Netze flexible und

robuste Lösungen mit verbesserter Funktionalität und ergänzen oder ersetzen die konventionellen Methoden mit ihrer Fähigkeit zu lernen und weitgehend selbständig eine Lösung zu finden.

Die Regelungstechnik befaßt sich mit der Aufgabe, ein sich zeitlich veränderndes System von außen gezielt so zu beeinflussen, daß die Veränderungen in einer gewünschten Weise ablaufen. Die Regelungstechnik liefert die Ausgangsbasis und Methoden für die Automatisierung technischer Prozesse. Für lineare Systeme gibt es heutzutage eine ausgereifte Theorie, die Behandlung nichtlinearer Systeme hingegen wirft noch zahlreiche Probleme auf.

Ein schwieriges Problem in der Regelungstechnik stellt der Umgang mit Totzeiten bzw. mit Totzeitsystemen dar. Ein System ist mit Totzeiten behaftet, wenn es mindestens eine Komponente enthält, deren Eingaben nicht sofort, sondern erst nach einer bestimmten Zeitverzögerung die Ausgaben beeinflussen. In realen Systemen kommen derartige Totzeiten häufig vor und sie erschweren die Systemregelung, besonders, wenn sie sich mit der Zeit ändern.

Der Totzeitproblematik wird bisher ungenügend Rechnung getragen. Bei der Modellierung eines Systems werden Totzeiten einfach wegabstrahiert, da sie schwierig zu modellieren sind, insbesondere bei nichtlinearen dynamischen Systemen in Forschung und Industrie. Der Grund für das Ignorieren der Totzeiten liegt darin, daß ihre Größen unbekannt sind und nicht identifiziert werden können, wodurch eine optimale Modellierung und Regelung des Systems verhindert wird. Eine Identifikation von Totzeiten bzw. zumindest ihrer Größenordnung wäre daher sehr gewinnbringend für die Modell- und Regelgüte. Forschungsbeiträge zu diesem Thema behandeln fast ausschließlich lineare Systeme. Eine Betrachtung nichtlinearer dynamischer Systeme unter Einsatz intelligenter Methoden, wie sie hier verfolgt wird, bildet daher einen wesentlichen Fortschritt auf diesem Gebiet.

Der Erfolg führt zum einen zu einer besseren Regelung und damit je nach Anwendungsszenario auch zu einer besseren Umweltverträglichkeit, z.B. durch Verminderung des Schadstoffausstoßes bei Verbrennungsmotoren. Zum anderen können verringerte Entwicklungskosten und eine schnellere Marktreife im Bereich von Neuentwicklungen erreicht werden - Aspekte, die am heutigen Markt wirtschaftlich von großer Bedeutung sind.

1.2 Zielsetzung

Die vorliegende Arbeit leistet einen Beitrag zur Lösung der Totzeitproblematik. Kurz gefaßt ist die Zielsetzung die Identifikation von Totzeiten in nichtlinearen dynamischen Systemen unter Verwendung spezieller, für diesen Zweck entwickelter, neuronaler Netze mit dem Ziel einer verbesserten Systemmodellierung und -regelung. Dabei soll für die Identifikation gelten:

Beliebige Totzeiten Es muß möglich sein, beliebige Totzeiten zu identifizieren. Im Speziellen soll es keine Einschränkung auf ganzzahlige Werte oder eine bestimmte Abtastrate geben.

Zuverlässigkeit Das Ergebnis der Identifikation muß zuverlässig sein, damit es weiterverwendet werden kann. Es ist also nötig, ein Bewertungskriterium für das Ergebnis zu liefern.

Vorwissen Grundsätzlich soll die Identifikation autonom und ohne Bedarf an Vorwissen ablaufen. Liegen jedoch Kenntnisse über die zu identifizierende Totzeit vor, so soll dieses Wissen leicht in die Methode integrierbar sein, um dieses optimal auszunutzen.

Es wird ein neuer Ansatz vorgestellt, der diese Rahmenbedingungen erfüllt und damit eine wichtige Basis für die Modellierung und Regelung von Totzeitsystemen schafft. Anhand von Anwendungsbeispielen wird der erfolgreiche Einsatz der entwickelten Methode gezeigt, zum einen zur Totzeitidentifikation und zum anderen zur verbesserten Regelung von Totzeitsystemen.

Der in dieser Arbeit vorgestellte neuronale Ansatz zur Totzeitidentifikation ist nicht beschränkt auf spezifische Einzelanwendungen, sondern ist generell anwendbar auf nichtlineare dynamische totzeitbehaftete Systeme, wie sie in den verschiedensten Bereichen auftreten, nicht zuletzt auch in chemischen und umwelttechnologischen Prozessen. Anwendungspotential ergibt sich grundsätzlich bei komplexen, hochdynamischen Prozessen, über die geringes oder gar kein Vorwissen vorliegt.

1.3 Aufbau der Arbeit

Es folgt ein Überblick über die weiteren Kapitel dieser Arbeit.

Kapitel 2: Eine große Herausforderung in einem interdisziplinären Projekt ist es, eine gemeinsame „Sprache“ zu finden, da jedes Fachgebiet seine eigene Begriffswelt besitzt. Dieses Kapitel skizziert daher die wesentlichen Grundlagen der Regelungstechnik und legt die Verwendung einzelner Begriffe für diese Arbeit fest. Es soll vor allem Lesern, die nicht mit der Regelungstechnik vertraut sind, den Einstieg in die Thematik der Arbeit erleichtern.

Kapitel 3: Aufbauend auf den Grundlagen in Kapitel 2 wird hier das Phänomen Totzeit erläutert. Der Definition von Totzeiten, Zeitverzögerungen und Totzeitsystemen folgt eine Aufstellung einiger typischer Beispielsysteme. Die mit Totzeiten verbundenen Probleme werden erarbeitet, und bisherige Lösungansätze diskutiert. Damit wird der Bedarf neuronaler Identifikationsmethoden motiviert.

Kapitel 4: Dieses Kapitel faßt die Grundlagen und die Begriffswelt der neuronalen Netze zusammen. Es werden wesentliche Aspekte, z.B. Lernverfahren, erläutert und Probleme bei der Verwendung neuronaler Netze diskutiert. Abschließend werden Einsatzmöglichkeiten neuronaler Netze in der Regelungstechnik beschrieben. Dieses Kapitel soll den Einstieg in die Neuro-Informatik erleichtern.

Kapitel 5: Betrachtet man die Identifikation von Totzeiten in nichtlinearen dynamischen Systemen als Lernproblem für neuronale Netze, so sind spezielle Netzarchitekturen und Lernverfahren notwendig, die das Auftreten der Totzeiten berücksichtigen können. In diesem Kapitel wird die Modellauswahl für die neuronale Identifikation von Totzeiten dargelegt. Dabei wird gleichzeitig ein Überblick über derzeit verfügbare Modelle gegeben.

Kapitel 6: Dieses Kapitel stellt ein neues Konzept zur Identifikation von Totzeiten vor. Es beruht im Kern auf einem speziell für diese Aufgabe entwickelten neuronalen Netz, dem EAT-Netz. Es wird zunächst das Gesamtkonzept erläutert. Danach wird das EAT-Netz in Architektur und Lernverfahren beschrieben. Es wird definiert, wie aus einem trainierten Netz eine Totzeitidentifikation gewonnen und wie die Zuverlässigkeit dieser Identifikation beurteilt wird. Es wird dargelegt, wie eine Identifikation beliebiger Totzeiten realisiert wird, bei der unter Umständen vorhandenes Vorwissen eingebracht werden kann.

Kapitel 7: In diesem Kapitel wird die Funktionstüchtigkeit des in Kapitel 6 vorgestellten Konzepts anhand von Simulationsergebnissen belegt. Die Daten stammen aus einer Stromregelung bei Zwischenkreisumrichtern und wurden mit dem Simulationswerkzeug SABER, einem Verhaltens- und Netzwerksimulator aus dem Bereich der Mikroelektronik, erzeugt. Als Stromregelungsverfahren wird das Dead-Beat-Verfahren in den zwei Varianten prädiktiv und nichtprädiktiv verwendet. Das Kapitel gibt zunächst eine Übersicht über Stromrichter. Daran anschließend werden die beiden Varianten des Dead-Beat-Verfahrens erläutert und schließlich die Identifikationsergebnisse vorgestellt und bewertet.

Kapitel 8: Dieses Kapitel zeigt anhand von zwei Anwendungsbeispielen, wie mit Hilfe des neu entwickelten Verfahrens eine verbesserte Regelung von Totzeitsystemen erreicht wird. Als erstes Beispiel dient eine komplexe Prüfstandsimulation mit einem Ottomotor als Prüfling. Das zweite Beispiel zeigt die Regelung der Dampfzufuhr bei einer Mischanlage, die bei der Herstellung von Lebensmitteln verwendet wird. Beide Anwendungen liegen als Simulationen in Matlab/Simulink vor.

Kapitel 9: Dieses abschließende Kapitel faßt die Inhalte der Arbeit zusammen und gibt einen Ausblick auf weitere industrielle Anwendungsmöglichkeiten der vorgestellten Identifikationsmethode, sowie auf Ansätze zu zukünftigen Forschungsprojekten in diesem Bereich.

Kapitel 2

Grundlagen der Regelungstechnik

Wie jedes Fachgebiet hat auch die Regelungstechnik eine eigene Begriffswelt. In diesem Kapitel werden die für die Arbeit notwendigen Grundlagen der Regelungstechnik skizziert, um das Umfeld der Arbeit zu verdeutlichen, und den Leser in das komplexe Gebiet der Regelungstechnik einzuführen.

Im täglichen Gebrauch taucht der Begriff der *Regelung* häufig auf. Ein allgegenwärtiges Beispiel ist die Temperaturregelung in den eigenen vier Wänden, in denen ein angenehmes Wohnklima herrschen soll, und dies möglichst ohne viel Eigenaufwand und in Bezug auf den Ressourcenverbrauch möglichst optimal. Was aber genau verbirgt sich hinter diesem Begriff Regelung und was unterscheidet ihn z.B. von der sogenannten *Steuerung* ?

Oft werden Begriffe und vor allem Notationen in der Literatur unterschiedlich verwendet. In einem interdisziplinären Projekt ist es außerdem notwendig, eine gemeinsame „Sprache“ zu finden. Daher verfolgt dieses Kapitel auch das Ziel, die Bedeutung der einzelnen Begriffe so festzulegen, wie sie in dieser Arbeit verwendet wird.

Für eine umfassende Behandlung der Regelungstechnik sei auf die zahlreichen Lehrbücher [4, 9, 34, 35, 63, 40, 89] verwiesen. Weiterhin führen die Dissertationen [8] und [71] aus interdisziplinärer Sicht in die Thematik ein, erstere aus dem systemtheoretischen Blickwinkel, zweitere hauptsächlich unter dem Aspekt der Modellbildung in der Regelungstechnik.

2.1 Grundbegriffe der Regelung

Kurz gesagt besteht die Aufgabe der Regelungstechnik darin, ein vorgegebenes System derart zu beeinflussen, daß ein gewünschter Zustand eintritt.

Das vorhandene, zu beeinflussende System wird als *Strecke* oder *Prozeß* bezeichnet. Deren Ausgangsgröße, die *Regelgröße* y, soll einem bestimmten zeitlichen Verlauf folgen. Dies kann auch bedeuten, daß diese Größe konstant gehalten werden muß. Die Vorgabe für die Regelgröße erfolgt durch die *Führungsgröße* w. Der tatsächliche Verlauf der Regelgröße geht aus Messungen am System hervor und wird als *Istwert* bezeichnet, der dem Wert der Führungsgröße, dem *Sollwert* , folgen soll. Der Soll-/Istwertvergleich definiert den *Regelfehler* e, aus welchem der *Regler* die *Stellgrößen* u bestimmt, welche die Regelstrecke in geeigneter Weise beeinflussen. Bezeichnend für die Regelung ist der geschlossene Wirkungskreis (closed loop) durch die Rückführung des Regelfehlers. Dadurch ist es möglich auf unvorhersehbare externe Einflüsse, die sogenannten *Störgrößen* z, zu reagieren. Ist diese Rückführung nicht gegeben und somit die Regelgröße nicht erfaßt und die Störgrößen nicht berücksichtigt, so spricht man von *Steuerung* (open loop). Kennzeichnend für die Steuerung ist also der offene Wirkungsablauf. Einsatzgebiete der Steuerung ergeben sich, wenn keine Störeinflüsse vorhanden sind, und das Streckenverhalten hinreichend bekannt ist. Abbildung 2.1 veranschaulicht das eben beschriebene Prinzip der Regelung.

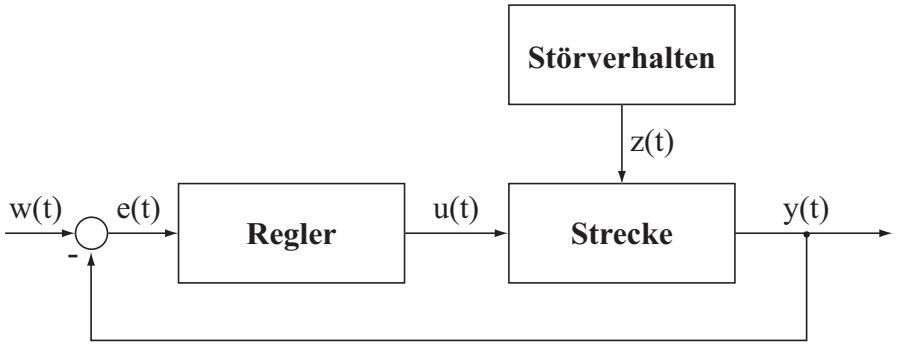

Abbildung 2.1: Veranschaulichung des Prinzips der Regelung nach [34]. Der Regler produziert aus dem Fehlersignal $e(t)$ das Stellsignal $u(t)$ für die Strecke. Wichtig ist dabei die Rückführung des Systemausgangs $y(t)$. Damit kann auf Störungen $z(t)$, die auf die Strecke einwirken, reagiert werden. Ohne diese Rückführung läge eine Steuerung vor. Es gibt auch Regler, die anstelle von $e(t)$ direkt mit den Signalen $y(t)$ und $w(t)$ versorgt werden.

2.2 Systemanalyse

Bei der Systemanalyse geht es darum, die Eigenschaften und das Verhalten eines gegebenen Systems zu untersuchen. Es wird zwischen Strukturanalyse und technischer Analyse unterschieden [51].

Die Aufgabe der Strukturanalyse besteht darin, die Struktur eines Systems zu finden und

sie mit Hilfe analytischer und graphischer Verfahren zu beschreiben und damit eine Formulierung zu finden, die in ein Programm abgebildet werden kann.

Die technische Analyse beschäftigt sich mit der Identifikation von Parametern. Hierbei kann zwischen konventionellen und kognitiven Verfahren unterschieden werden. Erstere bauen auf mathematischen Modellen auf, den Differentialgleichungen. Die kognitiven Verfahren entstammen meist aus der Mathematik, Informatik oder Physik und zeichnen sich durch intelligente Komponenten aus. In diesen Kontext, nämlich dem Einsatz kognitiver Verfahren zur Parameteridentifikation, fällt die vorliegende Arbeit. In Abschnitt 4.4 wird dieser Aspekt näher untersucht.

2.2.1 Systemarten

Systeme lassen sich nach Merkmalen klassifizieren [34, 44, 1]. Jede Klasse zeichnet sich durch gemeinsame Grundeigenschaften aus, die für das Verhalten des Systems und dessen mathematische Beschreibung von Bedeutung sind.

Zunächst werden **statische** von **dynamischen** Systemen unterschieden. Im statischen Fall hängt die Systemausgabe zum Zeitpunkt t nur von der Systemeingabe zum gleichen Zeitpunkt t ab. Solche Systeme sind über algebraische Beziehungen beschreibbar und werden gedächtnislose Systeme genannt. Im dynamischen Fall hängt die Systemausgabe nicht nur von der aktuellen Systemeingabe, sondern auch von deren zeitlichem Verlauf ab. Man spricht hier auch vom sog. Systemzustand und daher von gedächtnisbehafteten Systemen. Weiterhin gibt es die folgenden Klassifizierungsmöglichkeiten:

zeitkontinuierlich – zeitdiskret: In zeitkontinuierlichen Systemen sind alle Variablen für alle Zeitwerte innerhalb eines Intervalls definiert. In zeitdiskreten Systemen sind alle Variablen nur zu bestimmten Zeitpunkten definiert. Zeitkontinuierliche dynamische Systeme werden durch Differentialgleichungen, zeitdiskrete dynamische Systeme durch Differenzengleichungen beschrieben.

zeitinvariant – zeitvariant: Ein System wird als zeitinvariant bezeichnet, wenn alle Parameter über die Zeit konstant sind. Im Gegensatz dazu sind Systeme zeitvariant, wenn ein oder mehrere Parameter sich während des Betriebs ändern. Zum Beispiel hängt das Temperaturverhalten eines Schmelzofens davon ab, wie weit er mit Material gefüllt ist. Das Drehverhalten eines Motors ändert sich mit dessen Belastung. Zeitinvarianz bedeutet konstante Koeffizienten in Differentialgleichungen.

linear – nichtlinear: Dies ist eines der wesentlichsten Unterscheidungsmerkmale in der Regelungstechnik. Ein Operator L ist linear, wenn er dem Superpositions- und Verstärkungsgesetz folgt, welche zusammengefaßt wie folgt beschrieben werden können: für Funktionen g, f und h und für Skalare a und b gilt: wenn $h(t) = af(t)+bg(t)$, dann

$L(h) = aL(f) + bL(g)$. Ein System ist linear, wenn alle seine Komponenten[1] lineares Verhalten aufzeigen. Folgt mindestens eine Komponente nicht den Linearitätsgesetzen, so wird das System als nichtlinear bezeichnet. Für den linearen Fall existiert eine zufriedenstellende und ausgereifte Systemtheorie, und es stehen umfangreiche Werkzeuge zur Modellierung und Analyse zur Verfügung. Im nichtlinearen Fall hingegen sind Lösungen nur zum Teil bzw. nur für Spezialfälle möglich. Dies liegt darin begründet, daß es eine Vielzahl unterschiedlicher nichtlinearer Beziehungen gibt und es nicht möglich ist, mit allgemeinen Modellen alle Arten von Nichtlinearitäten zu beschreiben. Vielmehr muß jedesmal je nach Anwendung neu entschieden werden, welcher Ansatz zu wählen ist. Der nächste Abschnitt beschäftigt sich mit dieser Problematik.

2.2.2 Umgang mit Nichtlinearitäten

Im Folgenden werden verschiedene Möglichkeiten diskutiert, wie bei der Modellierung von Systemen mit Nichtlinearitäten umgegangen werden kann.

Linearisierung. Bei dieser Methode wird ein repräsentativer Arbeitspunkt des Systems gesucht, und mit Hilfe der Taylorreihenentwicklung die analytische Darstellung des Systems um diesen Arbeitspunkt linearisiert [34]. Dabei wird die Annahme zugrunde gelegt, daß die Abweichungen im Systemverhalten von diesem Arbeitspunkt gering sind. Das Hauptproblem bei dieser Methode ist das Auffinden des Arbeitspunktes, da er die Qualität der linearen Näherung bestimmt, die selbstverständlich hinreichend gut sein sollte. Doch selbst wenn der optimale Arbeitspunkt gefunden wird, darf nicht übersehen werden, daß das nichtlineare System stark vereinfacht wird zu einem engen Betriebsbereich um den gewählten Arbeitspunkt.

Erstellen eines Kennfelds. Hier wird versucht das System durch Messungen zu erfassen und dadurch ein Kennfeld zu erstellen [63]. Ein Kennfeld stellt die Abhängigkeit einer Systemgröße von einer oder mehreren anderen, meßbaren Systemgrößen dar. Im zweidimensionalen Fall spricht man von einer Kennlinie. Diese Methode wird meist dann angewandt, wenn der zugrundeliegende Prozeß nicht ausreichend bekannt ist. Allerdings ergibt sich hier das Problem, daß für eine hinreichend genaue Modellierung viele Messungen zu unterschiedlichen Betriebspunkten durchgeführt werden müssen, was einen hohen Aufwand sowie hohe Belastung des realen Systems bedeutet, insbesondere wenn Randpunkte im Betriebsbereich gemessen werden müssen.

Neutralisierung der Nichtlinearität. Bei diesem Ansatz wird davon ausgegangen, daß ein System grundsätzlich aus linearen und nichtlinearen Anteilen besteht. Das Gesamtsystem ergibt sich entweder durch Hintereinanderschalten dieser beiden Anteile,

[1] Jeder Komponente ist ein Operator zugeordnet. Ist der Operator linear, dann weist die zugehörige Komponente lineares Verhalten auf (vgl. dazu Abschnitt 2.2.3 auf S. 11).

oder durch deren additive Verschaltung. Bei der additiven Verschaltung wird ein Modell der Nichtlinearität subtraktiv aufgeschaltet und damit der nichtlineare Anteil neutralisiert. Übrig bleibt eine lineare Strecke, die mit gängigen Methoden leicht beherrschbar ist. Bei der Hintereinanderschaltung der Anteile wird die Nichtlinearität dadurch neutralisiert, daß ein inverses Modell der Nichtlinearität vorgeschaltet wird. Diese Kopplung ergibt die Identitätsabbildung, und die resultierende Strecke ist wiederum linear. Abbildung 2.2 verdeutlicht diese beiden Ansätze.

Das Problem bei dieser Sichtweise ist offensichtlich das Auftrennen der linearen und nichtlinearen Charakteristik, sowie das Auffinden eines Modells für die Nichtlinearität. Zum einen muß die Regelstrecke ausreichend bekannt sein und zum anderen muß die Nichtlinearität isoliert, d.h. meßbar, sein, um ein Modell davon erstellen zu können. Oft sind die Nichtlinearitäten aber nicht zugänglich. T. Brychcy beschreibt in seiner Dissertation [8] ein neuronales Verfahren zur Identifikation solcher versteckter Nichtlinearitäten und weist auf die Probleme in diesem Zusammenhang hin.

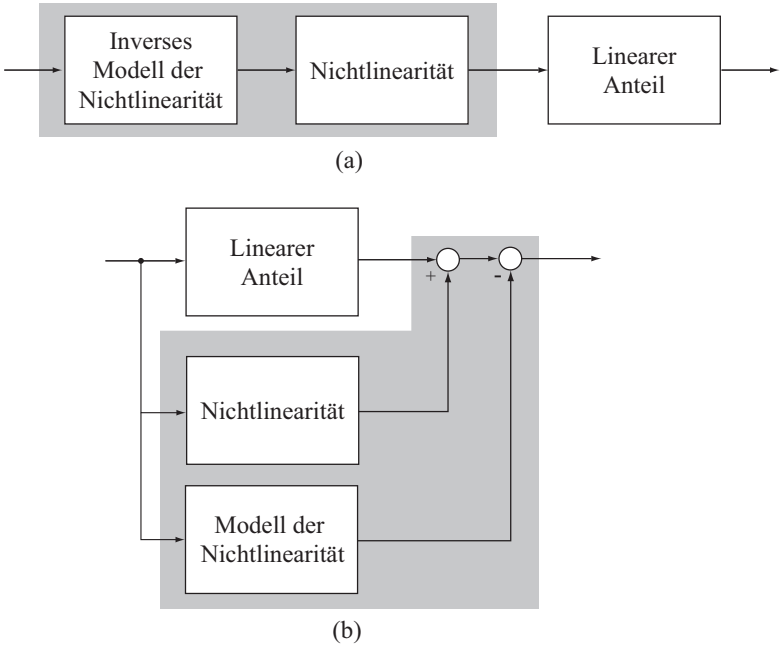

Abbildung 2.2: Neutralisierung von Nichtlinearitäten. Das System wird aufgeteilt in einen linearen und einen nichtlinearen Anteil. Bei der Hintereinanderschaltung (a) wird der Nichtlinearität ihr inverses Modell vorgeschaltet. Bei der additiven Verschaltung (b) wird ein Modell der Nichtlinearität subtraktiv aufgeschaltet. Die grau markierten Anteile ergeben jeweils die Identität. Insgesamt ergibt sich eine einfach beherrschbare lineare Gesamtstrecke. Diese Ansätze funktionieren allerdings nur dann, wenn ein Auftrennen in einen linearen und einen nichtlinearen Anteil möglich ist und dies ist leider selten der Fall.

Spezialfälle. Es gibt in der Regelungstechnik eine Reihe von Spezialfällen, darunter die sog. „harten“ Nichtlinearitäten, die nur über Spezialverfahren modellierbar sind [1]. Ein Beispiel hierfür ist die Hysterese, wie sie in Abbildung 2.3 zu sehen ist. Eine ausführliche Behandlung des Themas Identifikation von Hysterese befindet sich in der Dissertation [37]. Dort wird ein neues Verfahren mit Hilfe von neuronalen Netzen zur Identifikation von Systemen mit Hysterese vorgestellt.
Einen weiteren Spezialfall in der Regelungstechnik behandelt die vorliegende Arbeit. Totzeiten kommen zahlreich vor, sind aber schwer modellierbar. Kapitel 3 führt ausführlich in diese Thematik ein.

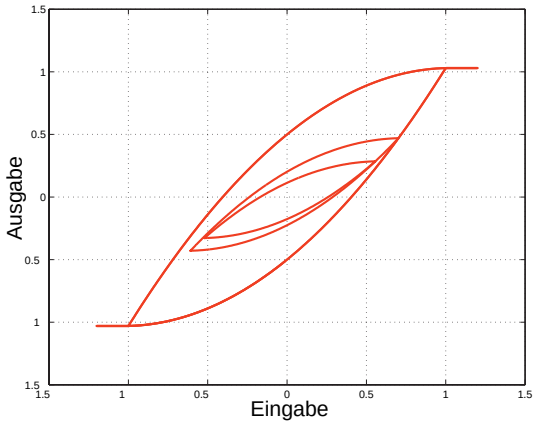

Abbildung 2.3: Beispiel einer Hysterese. Typisch ist die Ausbildung von Schleifen, wenn die Eingabe gegen die Ausgabe geplottet wird.

Zusammenfassend läßt sich feststellen, daß der Umgang mit Nichtlinearitäten einhergeht mit vereinfachenden Maßnahmen der Linearisierung, hohem Meßaufwand oder Anwendung problemorientierter Spezialverfahren. Steigende Qualitätsansprüche auf dem derzeitigen Stand der Technik fordern eine immer genauere Modellierung und damit auch die Berücksichtigung der Nichtlinearitäten.

An dieser Stelle sind neuronale Netze einsetzbar. Sie sind in der Lage Nichtlinearitäten zu modellieren, und ihre Parameter werden anhand von Meßdaten optimiert. Kapitel 4 führt näher in diese Thematik ein, und Abschnitt 4.4 diskutiert die Einsatzmöglichkeiten neuronaler Netze in der Regelungstechnik. Kapitel 6 zeigt den neuen Einsatz neuronaler Netze zur Identifikation von Totzeiten anhand von Messungen des Ein-/Ausgabeverhaltens eines Totzeitsystems.

2.2.3 Blockdiagramme

Bei der Beschreibung des Verhaltens eines Prozesses bedient sich die Regelungstechnik einer anschaulichen graphischen Darstellungsform, der Blockdiagramme [9, 34]. Abbildung 2.1 auf S. 6 zeigte bereits ein Blockdiagramm für das Prinzip der Regelung. An dieser Stelle wird diese Darstellungsform näher erläutert.

Ein Blockdiagramm besteht aus Blöcken, die den technischen Komponenten des dargestellten Prozesses entsprechen. Abbildung 2.4 zeigt den prinzipiellen Aufbau eines solchen Blockes. Jedem Block ist ein Operator zugeordnet, der die Gesamtheit aller Abbildungen angibt, die die Eingangsgröße auf die Ausgangsgröße abbilden und damit die Funktion bzw. Wirkung des Blockes beschreibt. Man spricht in diesem Zusammenhang auch von einer sog. Übertragungsfunktion. Pfeile im Blockdiagramm geben dabei das Zusammenspiel der einzelnen Blöcke an, d.h. welcher Block auf welche anderen Blöcke einwirkt. Dabei kann ein Blockdiagramm hierarchisch aufgebaut sein, d.h. ein Block kann selbst wiederum aus mehreren (Sub-)Blöcken bestehen. Aus den einzelnen Übertragungsfunktionen ergibt sich in zeit- und wertkontinuierlichen Systemen ein Differentialgleichungssystem, das den Gesamtprozess beschreibt. Ein Block mit zugeordneter Übertragungsfunktion wird auch als Übertragungsglied bezeichnet.

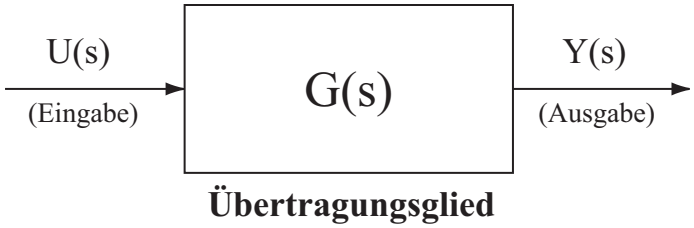

Abbildung 2.4: Prinzipieller Blockaufbau nach [34]. Die Übertragungsfunktion G(s) gibt an, wie die Eingabe U(s) auf die Ausgabe Y(s) abgebildet wird und es gilt: Y(s)=G(s)U(s).

Bei der Kennzeichnung des Verhaltens der Übertragungsglieder gibt es zwei Möglichkeiten. Die erste Möglichkeit ist die Angabe der Übertragungsfunktion als Laplace-Transformierte, wie in Abbildung 2.4 dargestellt. Die Laplace-Transformation $\mathfrak{L}$ ist ein handliches Werkzeug zur mathematischen Bearbeitung von Differentialgleichungen und entspricht einem Übergang vom Zeitbereich in den Frequenzbereich, auch Original- bzw. Bildbereich genannt. Für eine Funktion $f(t)$ der Zeit ergibt sich die Bildfunktion $F(s)$ der (komplexen) Variablen s durch folgende Definition:

$$\mathfrak{L}[f(t)] = \int_0^\infty f(t)e^{-st}dt = F(s) \tag{2.1}$$

Für die Transformation steht ein umfangreiches Tabellenwerk mit den gängigen Original- und den zugehörigen Bildfunktionen zur Verfügung, so daß die Transformation meist nicht explizit durchgeführt werden muß. Mit Hilfe einfacher Rechenregeln kann schnell aus dem Blockschaltbild auf das Verhalten des Gesamtsystems geschlossen werden. Die Laplace-Transformation gilt für kontinuierliche Systeme. Für diskrete Systeme gibt es die z-Transformation, welche häufig als diskrete Laplace-Transformation bezeichnet wird. Näheres zu diesen Transformationen zusammen mit den zugehörigen Tabellen befindet sich in den Lehrbüchern [14], [40], [34] und [35].

Von der z-Transformation kommt eine in der Regelungstechnik übliche Notation, welche in dieser Arbeit bei der Darstellung neuronaler Netze verwendet wird (vgl. Kapitel 5). Sei (x_i) mit $i \in \mathbb{N}$ eine Folge mit der z-Transformierten $X(z)$, dann hat die um einen Zeitschritt verschobene Folge $(w_i) = (x_{i-1})$ mit $w_1 = x_0 = 0$ die z-Transformierte $W(z) = z^{-1}X(z)$. Daher wird die Verzögerung um einen Zeitschritt mit dem Symbol z^{-1} bezeichnet.

Die zweite Möglichkeit der Kennzeichnung von Übertragungsgliedern ist die Verwendung der graphischen Darstellung des Antwortverhaltens der Ausgangsgröße auf eine bestimmte Veränderung der Eingangsgröße. Die *Sprungantwort* bezeichnet dabei die Reaktion der Ausgangsgröße auf eine sprunghafte Veränderung der Eingangsgröße mit anschließendem Halten des neuen Wertes. Meist handelt es sich hierbei um die Reaktion auf den sog. Einheitssprung, also das sprunghafte Ansteigen von Null auf Eins mit anschließendem Halten des Wertes Eins. Die *Impulsantwort* ist die Reaktion der Ausgangsgröße auf den Dirac-Impuls $\delta(t)$, ein infinitesimal kurzer Impuls zum Zeitpunkt $t = 0$ mit der Eigenschaft $\int \delta(t)dt = 1$. Die *Anstiegsantwort* stellt die Reaktion der Ausgangsgröße auf eine Eingabe mit konstantem Anstieg und Steigung Eins dar. Meist wird die Sprungantwort zur Kennzeichnung eines Blockes verwendet.

2.2.4 Zustandsraum

Eine weitere Möglichkeit, das Verhalten dynamischer Systeme zu beschreiben, ist die Zustandsraumdarstellung [89]. Sie besteht aus einer Zustandsgleichung für den Zustand $x(t)$ und einer Ausgangsgleichung für die Ausgangsgröße $y(t)$ zum Zeitpunkt t.

$$\dot{x}(t) = f(x(t), u(t)) \tag{2.2}$$
$$y(t) = g(x(t), u(t)) \tag{2.3}$$

Dabei gibt (2.2) an, mit welcher Geschwindigkeit sich der Zustand ändert, und (2.3) gibt den Zusammenhang zwischen der Ausgabe und dem Zustand an, beides in Abhängigkeit von der Eingabe. f und g sind vektorwertige Funktionen, welche die Art der Verknüpfung angeben. Sind f und g lineare, zeitinvariante Funktionen, so lassen sich (2.2) und (2.3) mit

geeigneten Matrizen A, B, C, D schreiben als

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2.4}$$

2.3 Regler

Wie in Abbildung 2.1 auf S. 6 verdeutlicht, ist der Regler Teil des Regelungskreises. Seine Aufgabe besteht darin, die Regelgröße y laufend mit der Führungsgröße w zu vergleichen, also den Regelfehler e festzustellen. Bei einem Regelfehler ungleich Null soll der Regler eine Stellgröße u liefern, so daß der Fehler beseitigt oder zumindest verringert wird.

Es gibt eine Vielzahl verschiedener Arten von Reglern [9, 63]. Es ist nun Aufgabe des Regelungstechnikers, für einen bestimmten Anwendungsfall einen geeigneten Regler auszuwählen und optimal einzustellen. Optimal eingestellt ist ein Regler, wenn die Regelgröße immer möglichst nah am Sollwert gehalten wird. Damit muß bei Änderung der Führungsgröße oder beim Auftreten von Störungen möglichst schnell reagiert werden.

Eine der häufig verwendeten Reglerarten ist der PID-Regler. Er besteht aus einer Parallelschaltung eines **P**roportional-, eines **I**ntegral- und eines **D**ifferential-Anteils. Mit dem Regelfehler e als Eingabe und der Stellgröße u als Ausgabe ergibt sich die Übertragungsfunktion für den PID-Regler im idealen Fall[2] als

$$u = K\left[e + \frac{1}{T_n}\int e + T_v\dot{e}\right] \tag{2.5}$$

Der PID-Regler besitzt drei Parameter: den Verstärkungsfaktor K (P-Anteil), die Nachstellzeit T_n (I-Anteil) und die Vorhaltezeit T_v (D-Anteil). Manchmal spricht man auch von den proportionalen, integralen bzw. differentiellen Verstärkungsfaktoren K, $\frac{K}{T_n}$ bzw. KT_v.

Der proportionale Anteil sorgt für schnelles Ausregeln der Regeldifferenz gegen Null. Er hat bei alleiniger Verwendung allerdings den Nachteil, daß aufgrund der Proportionalität von Ein- und Ausgangsgröße, eine Stellgröße nur dann vorhanden ist, wenn eine Regeldifferenz vorliegt — ist der Fehler gleich Null, so ist es auch die Stellgröße. Ist aber ein fester Anteil der Stellgröße $\neq$ Null zur Regelung nötig, so würde bei alleiniger Verwendung des P-Anteils immer eine Regeldifferenz bestehen bleiben. Abhilfe schafft hier der integrale Anteil, welcher die Geschwindigkeit berücksichtigt, mit der sich die Ausgangsgröße ändert. Er sorgt dafür, daß der Regelfehler immer vollständig ausgeregelt wird. Die

[2] Der ideale PID-Regler ist nicht realisierbar und nur von theoretischem Interesse. Probleme bereitet hierbei das Differenzieren. Daher wird der D-Anteil in der Praxis durch ein DT1-Anteil ersetzt. Näheres dazu findet sich in [34] oder [40].

Nachstellzeit T_n gibt dabei an, wieviel Zeit benötigt wird, um bei einer Sprungantwort den neuen Stellwert zu erreichen. Der differentielle Anteil bewirkt eine schnellere Ausregelung von starken Störungen. Die Vorhaltezeit T_v gibt dabei an, wieviel Zeit der proportionale Anteil brauchen würde, um die gleiche Störung auszuregeln.

Die drei Anteile des PID-Reglers ergänzen einander offensichtlich ideal im Sinne der schnellen und vollständigen Ausregelung von Regelfehlern. Die optimale Einstellung der Reglerparameter ist aber sehr schwierig, da die drei Parameter exakt aufeinander abgestimmt werden müssen. Daher wird oft auf den differentiellen Anteil verzichtet, und ein PI-Regler eingesetzt, der die Schnelligkeit des P-Anteils mit der Genauigkeit des I-Anteils kombiniert, ohne zu komplex einstellbar zu sein. Die Übertragungsfunktion eines PI-Reglers ergibt sich analog zu 2.5 als

$$u = K \left[e + \frac{1}{T_n} \int e \right] \tag{2.6}$$

Im Unterschied zum PID-Regler braucht der PI-Regler lediglich länger, um starke Störungen auszuregeln. Schätzungsweise werden derzeit in 95% aller Regelvorgänge PI oder PID-Regler verwendet [68].

2.4 Stabilität

Die Stabilität ist ein wichtiges Gütekriterium bei der Regelung [34]. Ein Regler, der das System in Dauerschwingungen versetzt, ist unbrauchbar. Die Regelgröße darf höchstens Schwingungen mit abklingender Amplitude ausführen und muß nach Beendigung des Einschwingvorgangs einen festen Wert erreichen. Diesen muß sie beibehalten, bis eine Änderung der Führungsgröße oder eine Störung eintritt. Ein solches System heißt stabil. Im wesentlichen heißt dies, daß ein stabiles System auf beschränkte Eingaben stets mit beschränkten Ausgaben reagiert. Ein System, das Dauerschwingungen durchführt, befindet sich an der Stabilitätsgrenze. Ein System mit aufklingenden Schwingungen ist dagegen instabil. Es sind eine Reihe von Verfahren zur Untersuchung der Stabilität bekannt, wie z.B. das Nyquist-Kriterium [40]. Dabei werden aber oft idealisierte Annahmen gemacht, wie z.B. die Linearität des Systems. Zwar kann argumentiert werden, daß nichtlineare Systeme vor dem Stabilitätsbeweis linearisiert werden können, jedoch gilt diese Aussage eben nur im betrachteten Arbeitspunkt und dort muss die instabile Charakteristik nicht zwingend vorhanden sein (vgl. Abschnitt 2.2.2).

Im Zusammenhang mit Totzeiten ist die Frage nach der Stabilität besonders interessant. Systeme mit Totzeiten neigen nämlich zu Schwingungen mit steigender Frequenz und Amplitude. Für die Analyse dieser Systeme ist daher die Kenntnis der Totzeit bzw. zumindest ihrer Größenordnung von starkem Interesse. Kapitel 3 diskutiert das Phänomen Totzeiten

ausführlich. Näheres zum Thema Stabilität von Totzeitsystemen findet sich in [44], [22] und [65].

2.5 Modellierung und Identifikation

Unter Modellierung versteht man den Vorgang des Auffindens eines Modells. Ein Modell bezeichnet dabei eine möglichst gute Approximation eines real existierenden Systems. Möglichst gut bedeutet dabei, daß Simulationen am Modell möglichst das gleiche Verhalten aufweisen sollen, wie es am realen System durch Messungen festgestellt werden kann. Durch die Modellierung technischer Systeme möchte man sich die teuren und die Anlage belastenden Messungen am realen System ersparen. Außerdem können am Modell Grenzfälle simuliert werden, ohne daß reale Hardware Schaden nimmt. Konventionelle Modelle in der Regelungstechnik sind meist Differentialgleichungen, die aufgrund physikalischer Zusammenhänge aufgestellt werden. Kognitive Modelle sind z.B. Verfahren aus dem Bereich der Fuzzy Logik oder neuronale Netze.

Im ersten Schritt der Modellierung muß eine Modellklasse gewählt werden, die geeignet erscheint, das zu modellierende System zu beschreiben. Im zweiten Schritt erfolgt die Identifikation der frei wählbaren Modellparameter, d.h. die Bestimmung geeigneter Belegungen für die Modellparameter. Dieser zweite Schritt wird als Systemidentifikation bezeichnet. Je komplexer ein System ist, desto mehr Genauigkeitsanforderungen müssen erfüllt werden und desto anfälliger ist das Gesamtsystem gegenüber Parameterveränderungen. Daher ist es erstrebenswert, ein Modell mit möglichst wenig zu identifizierenden Parametern zu erstellen. Anders ausgedrückt bedeutet dies, so viel Vorwissen wie möglich in das Modell einzubringen.

Bei der Systemidentifikation wird unterschieden, ob diese on-line oder off-line durchgeführt wird [47]. Off-line bedeuted, daß Meßdaten vom System gesammelt werden und diese später zur Systemidentifikation verwendet werden. Die Meßdaten stellen eine zeitinvariante Abbildung des Systems dar. On-line hingegen bedeutet, daß die Systemidentifikation im laufenden Betrieb abläuft und somit in das System selbst eingebunden ist.

Die Modellauswahl hängt selbstverständlich davon ab, welches System zugrundeliegt. Eine gute Modellierung und Identifikation muß dabei möglichst alle Eigenschaften des Systems berücksichtigen, um die steigenden Anforderungen an die Regelgüte erfüllen zu können.

Wie bereits erwähnt, handelt es sich bei Totzeiten um ein schwieriges Teilproblem in der Regelungstechnik. Das nächste Kapitel beschäftigt sich mit diesem Phänomen und seiner Problematik. Es werden bisherige Lösungansätze diskutiert, und damit das in dieser Arbeit verfolgte Ziel motiviert, mit neuronalen Modellen Totzeiten zu identifizieren.

Kapitel 3

Das Phänomen Totzeit

Systeme mit Totzeiten treten in den verschiedensten Anwendungsgebieten der Regelungstechnik auf. Zu den typischen Ursachen von Totzeiten zählen der Transport von Masse oder Energie, Übertragung von Information oder andere zeitbeanspruchende Prozesse. Bereiche, in denen Totzeiten eine wichtige Rolle spielen, reichen von Verbrennungsprozessen, Verfahrenstechnik, über Biologie, Medizin bis hin zur Ökonomie [44, 22, 38]. Charakteristisch für Totzeitsysteme ist die dem System anhaftende Trägheit.

3.1 Definition

Eine Totzeit bzw. ein Totzeitglied ist ein dynamisches System, welches das Eingangssignal $u(t)$ um die Zeit τ verzögert und durch die folgende Differenzengleichung beschrieben werden kann [56, 4].

$$y(t) = u(t - \tau) \tag{3.1}$$

Die Totzeit τ kann zeitvariant sein, d.h. eine Funktion der Zeit: $\tau = \tau(t)$. Charakterisiert wird ein Totzeitglied durch seine Sprungantwort, die in Abbildung 3.1 zu sehen ist.

Der Zustand des Systems (3.1) zum Zeitpunkt t, d.h. diejenige Information, die für die Berechnung des Ausgangssignals $y(t)$ gespeichert werden muß, wird durch den vergangenen Verlauf von $u(t)$ im Intervall $[t - \tau, t]$ beschrieben. Da ein Totzeitglied somit die gesamte letzte (kontinuierliche) Eingabetrajektorie der Länge τ zwischenspeichert, hat es theoretisch einen unendlich großen Zustand. In der Praxis ergibt sich durch die zeitdiskrete Verarbeitung am Rechner allerdings ein endlicher Zustand, dessen Größe von der zeitlichen Diskretisierung abhängt. Für die Berechnung des Ausgangssignals $y(t)$ wird eine Anfangsbedingung bzw. ein Anfangszustand benötigt, der nach [56] unter Verwendung einer geeignet gewählten Funktion φ wie folgt angegeben werden kann: $u(t + \rho) = \varphi(\rho),\ \rho \in [-\tau, 0]$.

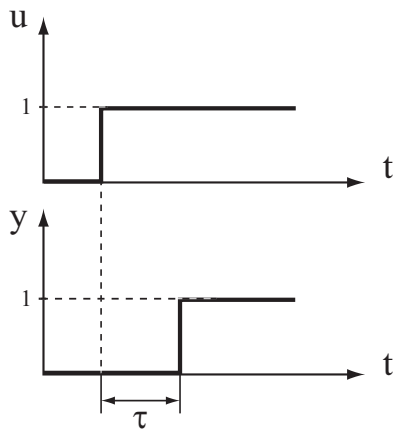

Abbildung 3.1: Sprungantwort eines Übertragungsglieds mit Totzeitverhalten. Das Diagram zeigt die Reaktion eines reinen Totzeitglieds auf den Einheitssprung. Nach Ablauf der Zeit τ erscheint am Ausgang das gleiche Signal wie am Eingang.

Bei Anwendung des Verschiebungssatzes der Laplace-Transformation auf (3.1) ergibt sich (vgl. [4])

$$Y(s) = e^{-\tau s} U(s) \tag{3.2}$$

Daher lautet die Laplace-Transformierte der Übertragungsfunktion eines Totzeitglieds $G(s) = e^{-\tau s}$.

Die Gleichung (3.1) bezeichnet dabei das sog. reine Totzeitglied — nach Ablauf der Totzeit τ erscheint am Ausgang genau das gleiche Signal wie am Eingang. Häufig tritt eine Totzeit nicht klar abgegrenzt auf. Es kann auch zu einer „nicht ganz echten Totzeit“ kommen [36]; in diesem Fall spricht man von einer Verzögerungszeit oder Zeitverzögerung. Dies tritt vor allem bei Mischvorgängen auf, wie z.B. bei einer Warmwasserheizung. Verursacht werden Verzögerungen durch Energiespeicher, wie z.B. Kondensatoren, Spulen, Massen oder Federn. Nach der Anzahl der Energiespeicher werden Verzögerungsglieder erster, zweiter und höherer Ordnung unterschieden. Diese entspricht der Ordnung des zugrundeliegenden Differentialgleichungssystems. Das sog. T_1-Glied ist ein Verzögerungsglied erster Ordnung. Die Sprungantwort eines T_1-Gliedes läßt sich berechnen mit

$$y(t) = (1 - e^{-\frac{t}{T_1}})\, u(t) \tag{3.3}$$

Die Zeitkonstante T_1 entspricht der beim Ladeverhalten von Kondensatoren auftretenden Konstanten, dem Produkt aus Kapazität und Parallelwiderstand: $T_1 \hat{=} R \cdot C$. Nach der Zeit T_1 hat die Ausgangsgröße 63% ihres Endwertes erreicht (vgl. Abbildung 3.2).

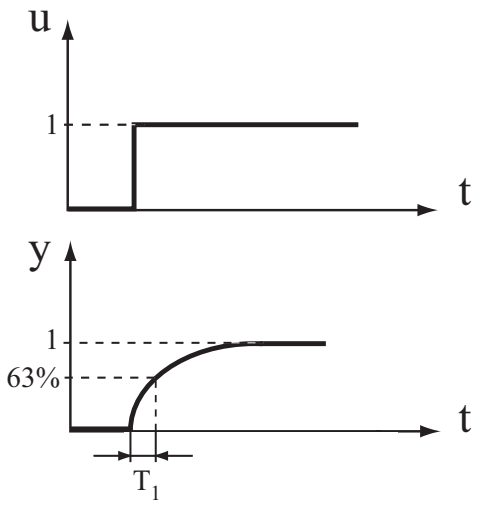

Abbildung 3.2: Sprungantwort eines T_1-Glieds. Auch hier wirkt sich der Eingang zeitverzögert auf den Ausgang aus. Nach Ablauf der Zeit T_1 hat die Ausgangsgröße 63% ihres Endwertes erreicht

Bei Verzögerungsgliedern zweiter und höherer Ordnung kann mit Hilfe der Sprungantwort eine Ersatztotzeit angegeben werden, so daß auch hier von Totzeitverhalten gesprochen werden kann [63]. Abbildung 3.3 verdeutlicht diesen Zusammenhang.

In dieser Arbeit werden die beiden Begriffe Totzeit und Zeitverzögerung äquivalent verwendet und bezeichnen das Phänomen, daß sich der Eingang einer Komponente *zeitverzögert* auf den Ausgang auswirkt.[1] Ein System wird als Totzeitsystem bezeichnet, wenn es mindestens eine Komponente enthält, die dieses Verhalten aufzeigt.

Das dynamische Verhalten von Totzeitsystemen wird mit Hilfe von Funktional-Differentialgleichungen (functional differential equations, FDE) beschrieben. Gewöhnliche Differentialgleichungen (ordinary differential equations, ODE) setzen eine unbekannte Funktion in Beziehung zu deren Ableitungen und zwar für ein und dasselbe Argument. Die Gleichung $\dot{x}(t) = 2x(t)$ beispielsweise stellt eine gewöhnliche Differentialgleichung mit Argument t dar. Anders ausgedrückt bildet bei gewöhnlichen Differentialgleichungen die unabhängige Variable (in diesem Fall t) das Argument der abhängigen Variablen (in diesem Fall x) und deren Ableitungen. Funktional-Differentialgleichungen hingegen verknüpfen eine unbekannte Funktion mit deren Ableitungen für unterschiedliche Argumente. Beispielsweise stellt die Gleichung $\dot{x}(t) = x(t+1) - [x(t-2)]^2$ eine FDE mit den Argumenten t, $t+1$ und $t-2$ dar. Ist die unabhängige Variable t die Zeit, so tritt die abhängige Variable also zu unterschiedlichen Zeiten auf.

[1] Im Englischen wird der Ausdruck *time delay* verwendet.

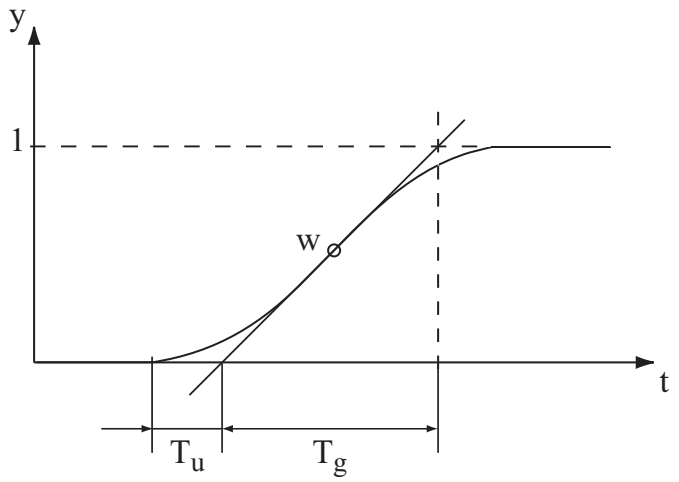

Abbildung 3.3: Sprungantwort eines T_n-Glieds. Nach Anlegen der Wendetangente können die Ersatztotzeit (Verzugszeit) T_u und die Ausgleichszeit T_g ermittelt werden.

In der Zustandsraumdarstellung ist ein kontinuierliches Totzeitsystem nach [44] beschrieben durch die Zustandsgleichung

$$\begin{aligned} \dot{x}(t) &= f(x(t), x(t-\tau_{x1}), x(t-\tau_{x2}), \ldots, x(t-\tau_{xN}), \\ &\quad u(t), u(t-\tau_{u1}), u(t-\tau_{u2}), \ldots, u(t-\tau_{uR}), t) \end{aligned} \tag{3.4}$$

und die Ausgangsgleichung

$$\begin{aligned} y(t) &= g(x(t), x(t-\tau_{x1}), x(t-\tau_{x2}), \ldots, x(t-\tau_{xN}), \\ &\quad u(t), u(t-\tau_{u1}), u(t-\tau_{u2}), \ldots, u(t-\tau_{uR}), t) \end{aligned} \tag{3.5}$$

mit $\tau_{xi}, \tau_{uj} \in \mathbb{R}^+, i = 1, 2, \ldots, N \in \mathbb{N}, j = 1, 2, \ldots, R \in \mathbb{N}$. Bei zeitinvarianten Systemen hängen die Funktionen f und g nicht explizit von t ab.

Im linearen Fall (f und g sind lineare Funktionen) ergibt sich die nachfolgende Darstellung.

$$\begin{aligned}\dot{x}(t) &= A_0(t)x(t) + \sum_{i=1}^{N} A_i(t)x(t-\tau_{xi}) \\ &\quad + B_0(t)u(t) + \sum_{i=1}^{R} B_i(t)u(t-\tau_{ui}) \qquad (3.6)\\ y(t) &= C_0(t)x(t) + \sum_{i=1}^{N} C_i(t)x(t-\tau_{xi}) \\ &\quad + D_0(t)u(t) + \sum_{i=1}^{R} D_i(t)u(t-\tau_{ui}) \qquad (3.7)\end{aligned}$$

Die Matrizen $A_i, C_i, i = 0, 1, \ldots, N$ und $B_i, D_i, i = 0, 1, \ldots, R$ sind konstant im Fall der Zeitinvarianz.

Für diskrete, lineare Systeme mit fester Abtastrate und ganzzahligen Vielfachen der Abtastrate als Verzögerungen ergeben sich analog zu (3.4) und (3.5) reine Differenzengleichungen

$$\begin{aligned}x(k+1) &= f(x(k), x(k-1), x(k-2), \ldots, x(k-N), \\ &\quad u(k), u(k-1), u(k-2), \ldots, u(k-R), k) \qquad (3.8)\\ y(k) &= g(x(k), x(k-1), x(k-2), \ldots, x(k-N), \\ &\quad u(k), u(k-1), u(k-2), \ldots, u(k-R), k) \qquad (3.9)\end{aligned}$$

Zu beachten ist, daß keine reinen Differenzengleichungen angegeben werden können, wenn die oben genannten Einschränkungen nicht gelten. Im Falle einer variablen Abtastrate oder beliebigen Totzeiten also ist die Darstellung wie in (3.4) und (3.5) zu wählen.

Die Differentialgleichung (3.4) gehört genaugenommen zur Klasse der verzögerten Funktional-Differentialgleichungen (retarded functional differential equation, RFDE). Eine ausführliche Betrachtung der Theorie der FDE geben [22], [24] und [38].

3.2 Beispielsysteme

Typische Vertreter von Totzeitsystemen sind Transportsysteme, in denen Masse oder Energie transportiert werden muß. Klassische Totzeitglieder sind hierbei Förderbänder oder Rohrleitungen. Im Falle der Rohrleitung errechnet sich die Totzeit aus der Länge des Rohres dividiert durch die Ausbreitungsgeschindigkeit des durchfließenden Materials [63]. Unter

Umständen sind zusätzlich Mischvorgänge oder Temperaturabhängigkeiten zu berücksichtigen, die sich auf die Totzeit auswirken. Ein Beispiel aus dem täglichen Umfeld ist hier die Gebäudeheizung. Auf eine Änderung der Zimmertemperatur kann erst zeitverzögert reagiert werden. Fällt beispielsweise die Raumtemperatur ab, so muß das Wasser im Kessel erhitzt werden und das heiße Wasser durch die Rohrleitung zu den Heizkörpern gelangen.

In eine ähnliche Kategorie fällt die Übertragung von Information. Hier hängt die Zeitverzögerung meist von der Last im Übertragungskanal ab. Große Entfernungen und Interaktionsanforderungen erschweren eine schnelle Regelung. Dies ist beispielsweise der Fall bei Anwendungen im Weltraum, wenn Sonden oder Roboter ferngesteuert werden müssen [18].

Ein weiteres Beispiel findet sich in der Automobilindustrie. Verbrennungsmotoren, wie der Ottomotor, stellen mit ihren komplexen dynamischen Verhalten eine anspruchsvolle Regelstrecke dar [56]. Zeitverzögerungen finden sich hier beispielsweise in der Strecke *Drosselklappe - Ansaugrohr - Verbrennungstakt* oder bei der zeitverzögerten Messung des Gas-/Luftgemisches im Motor [32]. In Abschnitt 8.1 werden Versuche mit einer Prüfstandsimulation beschrieben, die als Prüfling einen Ottomotor enthält. Dort findet sich auch eine genaue Beschreibung der Regelstrecke. Unter Verwendung der in dieser Arbeit entwickelten neuronalen Methoden wird eine verbesserte Regelung des Motors erreicht.

Einen weiterführenden Überblick über verschiedene Totzeitsysteme bieten [22] und [44]. Der nächste Abschnitt verdeutlicht, welche Probleme mit Zeitverzögerungen verbunden sind.

3.3 Totzeitproblematik

Wie bereits erwähnt, wirkt sich bei einem Totzeitglied eine Veränderung in der Eingabe nicht sofort, sondern zeitverzögert auf den Ausgang aus. Das heißt aber auch, daß Fehler im System zeitverzögert erkennbar werden und somit auch nur zeitverzögert darauf reagiert werden kann. Dies umfaßt auch den Fall, daß unter Umständen schlichtweg *zu spät* reagiert wird. Es ist außerdem möglich, daß sich Störungen durch Rückführungen im System mehrfach auswirken und sich dadurch verstärken. Man spricht in diesem Zusammenhang auch von Selbsterregung [84]. Totzeitsysteme neigen damit zu Instabilität, und insgesamt leidet die Regelgüte unter Umständen erheblich.

Es ist keineswegs übertrieben, zu sagen, daß Zeitverzögerungen in jedem (realen) System vorkommen. [56] beispielweise schreibt hierzu: „Regelstrecken haben in der Praxis immer zeitliche Verzögerungen“. Ähnlich argumentiert [44]: „It can be claimed that any real-life system has some time delay associated with it“. Das heißt natürlich nicht, daß in jedem erdenklichen System alle Zeitverzögerungen berücksichtigt werden müssen. Oft können sie

vernachlässigt werden, aber eben nicht immer. Vielmehr muß bei jeder Anwendung neu die Frage beantwortet werden, inwieweit Totzeiten berücksichtigt werden müssen.

Bei totzeitbehafteten Systemen interessiert den Regelungstechniker die Frage, inwieweit die Totzeiten das dynamische Verhalten der Regelstrecke beeinflussen. Maßgebend dafür sind die Größenordnungen der Totzeiten. Die Tatsache, daß eben diese meist unbekannt ist, bildet den Ansatzpunkt dieser Arbeit. Eine übliche Faustregel lautet, daß nur Totzeiten von mindestens der Größenordnung der dominanten Zeitkonstanten der Regelstrecke zu beachten sind. Aber bei hoher Regelgüte können die Zeitkonstanten des geregelten Systems deutlich kleiner als die der Regelstrecke werden, so daß die Totzeiten vielmehr mit der Dynamik des geregelten Systems zu vergleichen sind. Das bedeutet aber, daß bei hohen Regelqualitätsanforderungen bereits relativ kleine Totzeiten berücksichtigt werden müssen. Reglerauslegungen mit konventionellen Entwurfsmethoden, die diese Totzeiten nicht berücksichtigen, führen im allgemeinen zu schwingendem oder gar instabilem Verhalten [56]. Dies ist beispielsweise der Fall bei den in Abschnitt 2.3 eingeführten PI/PID-Reglern, welche keine Totzeiten berücksichtigen können. Abschnitt 8.2 zeigt eine Anwendung, die mit diesem Reglertyp nicht stabil zu regeln ist. Mit Hilfe der in dieser Arbeit entwickelten neuronalen Methode allerdings wird eine erfolgreiche Lösung erreicht.

Erschwerte Bedingungen herrschen bei zeitvarianten Totzeiten. Bei zeitinvarianten Totzeiten können vielleicht noch anhand von Messungen Erfahrungs- oder Richtwerte gefunden werden und daran entsprechende Regelverfahren ausgerichtet werden. Ändert sich die Totzeit aber im Betrieb, so muß auch das Regelverfahren adaptiv sein, da ansonsten wieder die Gefahr der Instabilität auftritt.

In vielen Anwendungen ist zwar bekannt, daß Totzeiten im System auftreten, jedoch sind diese gänzlich unbekannt und meßtechnisch nicht erfaßbar, d.h. es kann nicht direkt vor und hinter dem die Totzeit verursachenden Übertragungsglied gemessen werden. Hier sind Verfahren nötig, die allein aus dem Ein-/Ausgabeverhalten des Gesamtsystems eine Näherung für die Totzeit geben können. In dieser Arbeit wird daher ein neues Konzept zur Totzeitidentifikation auf Basis neuronaler Netze vorgestellt (vgl. Kapitel 6). Der nächste Abschnitt stellt bisherige Lösungsansätze zum Umgang mit Totzeiten dar.

3.4 Diskussion bisheriger Lösungsansätze

Die Probleme bei der Regelung totzeitbehafteter Systeme sind zwar bekannt, aber bis heute kaum gelöst. Dies liegt nicht zuletzt daran, daß Totzeiten nicht invertierbar sind — man müßte in die Zukunft blicken können. Entsprechend ist ein intuitives Aufstellen des Regelgesetzes praktisch nicht möglich.

Die einfachste Methode ist die „Vogel-Strauß-Methode“, d.h. „Kopf in den Sand stecken“ und so tun, als gäbe es kein Problem. Ein Totzeitsystem wird dann durch Ignorieren der

Totzeiten zu einem totzeitfreien System vereinfacht und mit klassischen Reglern (PI oder PID, vgl. Abschnitt 2.3) geregelt. Die Parameter der Regler können so eingestellt werden, daß sich ein langsames Ansprechverhalten ergibt. Dadurch erhofft man sich eine Überbrückung der Totzeit. Dies widerspricht jedoch der Anforderung einer schnellen Regelung. Wie bereits erwähnt wird auch die Forderung nach einer stabilen Regelung in den meisten Fällen nicht erfüllt. Aufgrund mangelnder Identifikationsmöglichkeiten gab es aber bisher keine andere Möglichkeit.

Den theoretisch idealen Regler für lineare Totzeitsysteme entwickelt P. M. Frank in [20]. Ausgehend von einer Strecke der Form $S(s) = S_1(s) \cdot e^{-\tau s}$ mit Totzeit τ wird die Übertragungsfunktion des idealen Reglers angegeben als

$$R(s) = \frac{R_\infty}{S(s)} = \frac{R_\infty}{S_1(s)}\, e^{+\tau s} \tag{3.10}$$

wobei $R_\infty \to \infty$ den Verstärkungsfaktor des idealen Reglers bezeichnet. Der Term $e^{+\tau s}$ drückt die Vorhersage der Regelabweichung um die Totzeit τ aus und ist physikalisch nicht realisierbar. Die optimale Regelung von Systemen mit Totzeit ist damit ein Vorhersageproblem.

Der in der Praxis weitestverbreitete Regler für Totzeitsysteme ist der Smith-Prädiktor [69], [22]. Er eignet sich aber nur für lineare Strecken, bei denen eine Totzeit in Serie mit einem stabilen totzeitfreien Prozeß auftritt. Gegebenenfalls sind also vereinfachte Annahmen über den zugrundliegenden Prozeß zu treffen. Gegeben sei ein linearer totzeitfreier Prozeß mit der Übertragungsfunktion $G_1(s)$. Mit einer in Serie geschalteten Totzeit τ ergibt sich als Gesamtsystem

$$G_0(s) = e^{-s\tau}\, G_1(s) \tag{3.11}$$

Für die lineare totzeitfreie Strecke $G_1(s)$ kann mit gängigen Methoden leicht ein Regler $G_R(s)$ konstruiert werden. Das Smith-Prinzip kombiniert diesen Regler mit einem Modell der Strecke $G_1(s)$ und einer Kompensation der Totzeit wie in Abbildung 3.4 verdeutlicht. Dieses Verfahren verlangt jedoch nicht nur ein genaues Modell der Strecke, sondern insbesondere die genaue Kenntnis der Totzeit, und es reagiert sehr empfindlich gegenüber Totzeitunterschieden zwischen Strecke und Modell.

Bei der Approximation von Totzeiten gilt die Padé-Approximation als die geeignetste Methode [35]. Dabei wird ein Totzeitglied durch ein Allpaßglied angenähert. Die Bezeichnung Allpaß kommt dabei von der Frequenz-Betrachtung: ein Allpaß läßt Sinusschwingungen beliebiger Frequenz ohne Amplitudenänderung passieren. Das prinzipiell unendlichdimensionale Totzeitglied wird durch ein endlichdimensionales reell-rationales Polynom

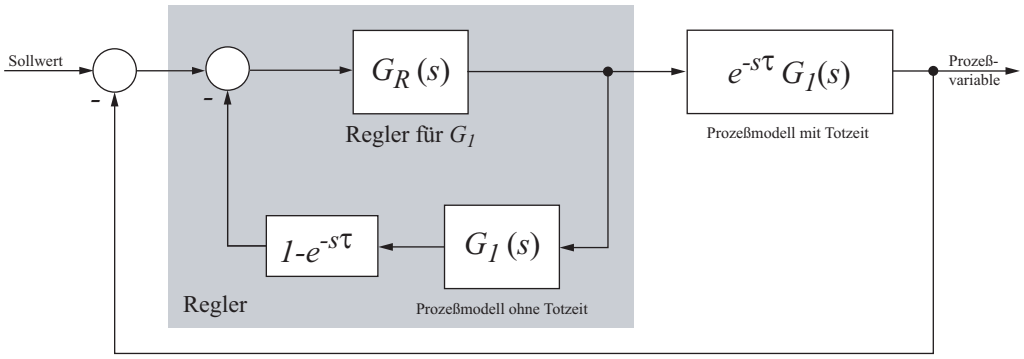

Abbildung 3.4: Regelung einer linearen Totzeitstrecke nach dem Prinzip von Smith. Der Regler für die Gesamtstrecke $G_0(s) = G_1(s)\ e^{-s\tau}$ besteht aus mehreren Komponenten und ist modellgestützt. Für die lineare totzeitfreie Strecke $G_1(s)$ wird ein Regler $G_R(s)$ konstruiert und in der Rückkopplung die Totzeit berücksichtigt.

mit Zählergrad M und Nennergrad N angenähert.

$$G(s) = e^{-\tau s} \approx \frac{\sum_{k=0}^{M} b_k \cdot (-s\tau)^k}{\sum_{k=0}^{N} a_k \cdot (s\tau)^k} \tag{3.12}$$

mit $b_k = \frac{(M+N-k)!M!}{k!(M-k)!}$ und $a_k = \frac{(M+N-k)!N!}{k!(N-k)!}$.

Bei Verwendung eines Allpasses 2. Ordnung ergibt sich $G(s) = e^{-\tau s} \approx \frac{(s\tau)^2-6(s\tau)+12}{(s\tau)^2+6(s\tau)+12}$. Das Problem ist, daß bei der Approximation in der Regel mit weit höheren Ordnungen angesetzt werden muß, um gute Ergebnisse zu erreichen.

Für lineare totzeitfreie Systeme ist mit der sogenannten H_∞-Methode eine bewährte und systematisch anwendbare Methode zum Reglerentwurf entwickelt worden [19, 17, 12]. Es handelt sich um ein mathematisches Verfahren, mit dessen Hilfe robuste Regelsysteme ausgelegt werden können. In der Dissertation [56] findet sich ein Ansatz zur Erweiterung der H_∞-Methode auf lineare totzeitbehaftete Systeme. Verifiziert wird das entwickelte Entwurfsverfahren anhand der Regelung eines Ottomotors. Dabei wird das dynamische Verhalten des Ottomotors um einen Betriebspunkt linearisiert, um zum gewünschten linearen Totzeitsystem zu gelangen.

Auch M. Jankovic und I. Kolmanovsky beschäftigen sich in [32] mit Verbrennungsmotoren und zeigen eine Methode zur Regelung des Gas-/Luftgemisches. Dazu wird das nachfolgende Modell eines Verbrennungsmotors verwendet.

$$\begin{aligned} \dot{P} &= c_m(4\theta\sqrt{P - P^2} - W_c) \\ W_c &= 30P\frac{N}{1000} \\ \lambda &= \frac{W_c}{W_f} \end{aligned} \tag{3.13}$$

Dabei bezeichnet W_f den Kraftstoffstrom, W_c den Luftmassenstrom, P den Ansaugdruck und N die Motordrehzahl. Das Gas-/Luftverhältnis λ wird zeitverzögert am Motorausgang gemessen: $\lambda_m(t) = \lambda(t - \tau)$. Zeitverzögerungen treten auf von der Kraftstoffeinspritzung bis zum Ausschubtakt und beim Transport vom Ausschubventil zum λ-Sensor. Die Zeitverzögerung ist abhängig von der Motordrehzahl N: $\tau = \tau(N)$. Als Regelverfahren wird in diesem Fall die Gain-Scheduling-Methode [60] angewandt. Das System wird in mehreren Arbeitspunkten linearisiert und lokal lineare Regler für jeden Arbeitspunkt entworfen, die derselben Reglerklasse angehören. Je nach Betriebspunkt werden die Parameter eines lokallinearen Reglers übernommen oder zwischen Parametern mehrerer Regler geeignet interpoliert. Die Bezeichnung Gain-Scheduling bezieht sich auf den englischen Ausdruck „gain“, der die Parameter eines Reglers bezeichnet (Verstärkungsfaktoren). Hier tritt wieder das Problem der adäquaten Wahl der Arbeitspunkte auf. Ist außerdem das Netz der Arbeitspunkte nicht engmaschig genug, kann eine Interpolation unter Umständen unbrauchbar sein.

Die Totzeitproblematik am Beispiel von Tracking- und Greifvorgängen im Weltraum behandelt C. Fagerer in [18]. Übertragungsverzögerungen vom Spacelab und zurück im Bereich einiger Sekunden sind hierbei zu berücksichtigen. Es wird vorgeschlagen, die Totzeit durch Prädiktion zu kompensieren. Allerdings werden auch hier vereinfachte Annahmen gemacht. Das System wird als linear und die Totzeit als ganzzahliges Vielfaches der Abtastzeit angenommen. Der Ansatz der Regelung durch Modellvorhersagen ist in der Literatur auch als *Model Predictive Control* (MPC) bekannt. Einen breiten Literaturüberblick bietet dazu [68]. Abschnitt 4.4 geht näher auf diese Thematik ein.

Zusammenfassend ist festzustellen, daß der bisherige Umgang mit Totzeitsystemen mit stark vereinfachenden Annahmen einhergeht. Entweder werden von vornherein nur lineare Systeme betrachtet oder durch Linearisierung dazu vereinfacht. Die Totzeiten selbst werden lediglich als ganzzahlige Vielfache der Abtastrate modelliert. Die Verwendung neuronaler Netze jedoch erlaubt es, nichtlineare Systeme in ihrer ganzen Komplexität zu betrachten. Im nächsten Kapitel werden zunächst die wesentlichen Eigenschaften neuronaler Netze eingeführt, bevor die darauf folgenden Kapitel den Einsatz neuronaler Netze zur Totzeitidentifikation erläutern.

Kapitel 4

Neuronale Netze

Künstliche neuronale Netze, in dieser Arbeit kurz neuronale Netze, sind als mathematische Modelle der biologischen Nervensysteme entstanden. Die Pionierarbeit auf diesem Gebiet leisteten 1943 der Neurologe W. McCulloch und der Mathematiker W. Pitts [45]. Wenig später legte D. Hebb den Grundstein der neuronalen Lernverfahren [26]. Heute bilden neuronale Netze ein breites, interdisziplinäres Forschungsgebiet, das Wissenschaftler aus den unterschiedlichsten Bereichen vereint. Interdisziplinäre Synergien entstehen bei der Entwicklung verschiedener neuronaler Modelle für unterschiedliche Anwendungsgebiete.

Dieses Kapitel faßt zunächst die Grundlagen neuronaler Netze zusammen und definiert, was unter dem Begriff Lernverfahren zu verstehen ist. Danach werden einige Probleme in der Anwendung neuronaler Netze diskutiert. Der letzte Abschnitt erläutert den bisherigen Einsatz neuronaler Netze in der Regelungstechnik.

Für eine umfassende Behandlung neuronaler Netze sind die Lehrbücher [57, 25, 13, 88] zu empfehlen.

4.1 Grundlagen

Neuronale Netze sind eine Abstraktion biologischer neuronaler Systeme. Sicherlich hat man sich zunächst zugunsten der mathematischen Modellierbarkeit weit entfernt vom biologischen Vorbild. Beispielsweise sind ableitungsbasierte Lernverfahren wie Backpropagation[1] mit ziemlicher Sicherheit nicht in der Biologie zu finden. Grundlegende Aspekte biologischer Nervensysteme jedoch werden mit künstlichen neuronalen Netzen modelliert. So besteht ein künstliches neuronales Netz wie sein natürliches Vorbild aus einer Menge von

[1] vgl. Abschnitte 4.2.2 und 5.1

einfachen, in der Regel in Schichten angeordneten Berechnungseinheiten, den Neuronen, die über gewichtete, veränderliche Verbindungen miteinander vernetzt sind und gemeinsam komplexe Aufgaben lösen können. Abbildung 4.1 zeigt den Aufbau eines biologischen Neurons.

Es besteht aus einem Zellkörper, der viele stark verzweigte Auswüchse, die sog. Dendriten, zeigt. Desweiteren gibt es einen langen Auswuchs, das Axon. Neuronen sind einfache Verarbeitungselemente, die von den Dendriten ankommende Eingaben im Zellkörper verarbeiten und das Ergebnis über das Axon weiterleiten. Das Axon ist über Synapsen an die Dendriten anderer Neuronen gekoppelt. In den Synapsen erfolgt mittels chemischer Transmitterstoffe die Signalübertragung [64, 57, 88].

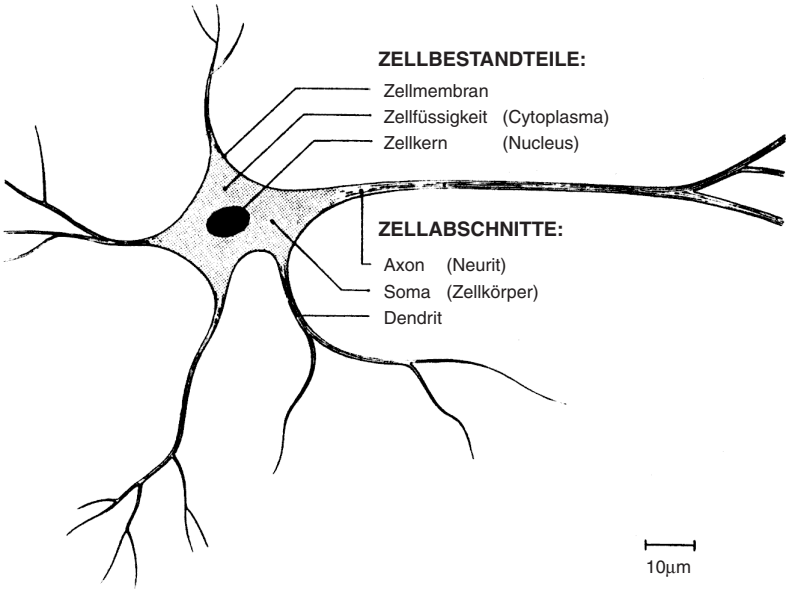

Abbildung 4.1: Aufbau eines biologischen Neurons nach [64]

Das künstliche Abbild übernimmt wie bereits erwähnt den prinzipiellen Aufbau eines Neurons. Die Funktionsweise wird jedoch stark vereinfacht modelliert. Die komplexe Wirkungsweise einer Synapse wird beispielsweise nur durch einen Gewichtswert dargestellt und der Verlauf des typischen Aktionspotentials durch eine sog. Aktivierungsfunktion angenähert. Abbildung 4.2(a) zeigt den Aufbau eines künstlichen Neurons. Die Eingaben u_i werden gewichtet mit den Gewichtswerten w_i aufsummiert. Mit der gewichteten Summe als Eingabe berechnet die Aktivierungsfunktion f die Ausgabe y. Die Berechnung des Ein-/Ausgabeverhaltens eines einzelnen Neurons läßt sich also angeben als

$$y = f\big(\sum_{i=1}^{N} w_i\, u_i\big) \tag{4.1}$$

Im weiteren Verlauf wird ein Neuron vereinfacht lediglich als ein Kreis dargestellt, der mit einem Symbol für die Aktivierungsfunktion gekennzeichnet ist. Meist werden sigmoide oder lineare Aktivierungsfunktionen verwendet, vgl. Abbildung 4.2 (b) und (c).

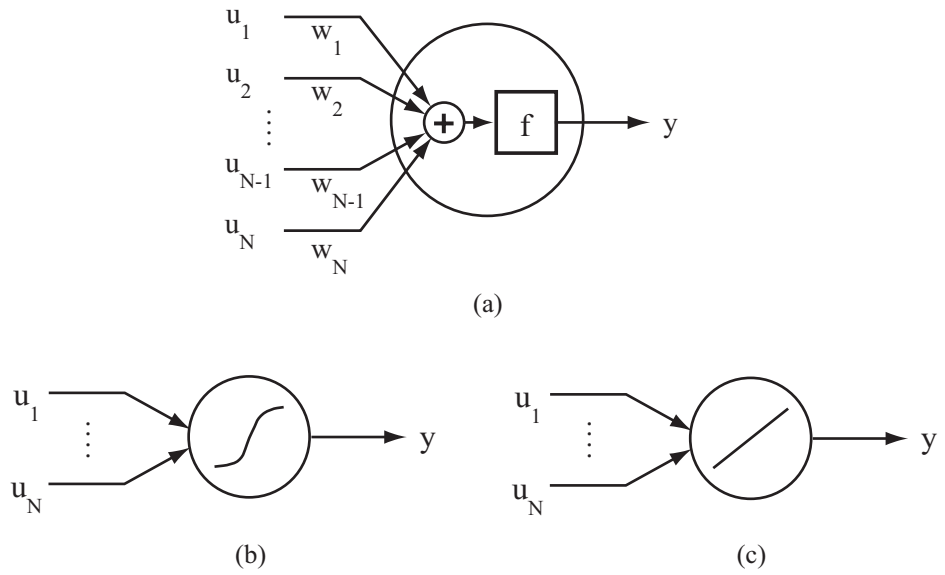

Abbildung 4.2: Aufbau und Darstellung eines künstlichen Neurons. (a) Die Eingaben u_i gelangen über mit w_i gewichtete Verbindungen ins Neuron. Die gewichtete Summe $\sum_{i=1}^{N} w_i\, u_i$ ist die Eingabe der Aktivierungsfunktion f, welche die Ausgabe y des Neurons berechnet. (b) Symbol für ein Neuron mit sigmoider Aktivierungsfunktion. (c) Symbol für ein Neuron mit linearer Aktivierungsfunktion.

Im Laufe der Zeit wurden viele unterschiedliche neuronale Modelle und Lernverfahren entwickelt. Dabei flossen auch Erkenntnisse und Methoden anderer Disziplinen in die Entwicklung ein. Das Prinzip des Optimierungsverfahrens *Simulated Annealing* (simulierte Vergütung) beispielsweise stammt aus der Metallurgie [57] und wird häufig bei neuronalen Lernverfahren verwendet. Das zunehmende Verständnis biologischer synaptischer Vorgänge führte zu vielen biologienahen neuronalen Modellen, welche beispielsweise bemüht sind, die synaptischen Vorgänge genauer zu modellieren. Die Regelungstechnik regte aus dem Bedarf der Identifikation dynamischer Systeme die Erweiterung eines Neurons um dynamische Komponenten an (vgl. Abschnitt 5.3.2).

In der Regel werden neuronale Netze nach ihrer Struktur klassifiziert. Die Struktur eines neuronalen Netzes ist bestimmt durch die Anzahl und Art der Neuronen und deren Verbindung untereinander. Dabei wird unterschieden, ob die Netze geschichtet bzw. ungeschichtet oder vorwärtsgerichtet bzw. rekurrent (d.h. rückgekoppelt) sind. Näheres dazu findet sich in Kapitel 5.

Die Festlegung einer geeignet erscheinenden Struktur bildet den ersten Schritt beim Einsatz neuronaler Netze. Dazu gehört auch die Auszeichnung trainierbarer Parameter. Dies sind meist die Gewichte des neuronalen Netzes. Je nach Modell sind aber auch weitere Parameter denkbar (vgl. Kapitel 5). Dabei sollte die Anzahl der trainierbaren Parameter möglichst gering gehalten werden, um eine gute Generalisierung zu erreichen und zu verhindern, daß die Daten vom Netz auswendig gelernt werden. Der Begriff Generalisierung bezeichnet dabei die Fähigkeit, von den vorgegebenen Lerndaten zu abstrahieren und auf den zugrundeliegenden Generierungsprozeß zu schließen. Auch bisher ungesehene Daten können dann vom neuronalen Netz korrekt verarbeitet werden. Durch Schätzung oder Zufallsinitialisierung erhält man initiale Belegungen der trainierbaren Parameter, die dann in einem Lernverfahren optimiert, d.h. so adaptiert werden, daß das neuronale Netz lernt, eine bestimmte Aufgabe zu lösen.

4.2 Lernverfahren

Das Lernverfahren bezeichnet die Vorgehensweise bei der Parameteradaption. Es gibt eine Vielzahl verschiedener Lernverfahren. Für die in dieser Arbeit betrachteten neuronalen Modelle (vgl. Kapitel 5 und 6) läßt sich das nachfolgende allgemeine Vorgehen beschreiben. Für über diese Arbeit hinausgehende Ansätze sei auf die zu Beginn dieses Kapitels aufgeführte Literatur verwiesen.

4.2.1 Allgemeines Vorgehen

Ein Lernverfahren neuronaler Netze ist ein iterativer Vorgang, der in dieser Arbeit als das *Training* neuronaler Netze bezeichnet wird. Jedes Lernverfahren setzt das Festlegen einer geeigneten Struktur für das zu trainierende Netz voraus. Dabei werden auch die trainierbaren Parameter ausgewählt und diese mit initialen Werten versehen. Dies kann durch Schätzung oder Zufallsinitialisierung in einem plausiblen Bereich geschehen. Ebenso wird eine Fehlerfunktion in den trainierbaren Parameter festgelegt, um die Performanz des Netzes zu bewerten. Aufgabe des Lernverfahrens ist es nun, diese Fehlerfunktion zu minimieren. Dies geschieht durch Optimierung der trainierbaren Parameter. Die Parameter zu optimieren, bedeutet diese so zu adaptieren, daß der Netzfehler kleiner wird. Ist das Ergebnis nach dem Training nicht zufriedenstellend, kann versucht werden mit einer anderen Zufallsinitialisierung oder gar mit einer veränderten Struktur und anschließendem erneuten Training ein besseres Ergebnis zu erzielen.

Das Lernverfahren benötigt zur Parameteroptimierung Messungen, die dem Netz als Eingaben $u_1, ..., u_n$ und als Ausgaben $y_1, ..., y_n$ zur Verfügung stehen. Ziel ist es, anhand dieser Messungen auf den Zusammenhang zwischen Ein- und Ausgaben zu schließen. Das Verfahren entspricht somit einem „Lernen aus Beispielen“ mit den gemessenen Datenpaaren

$(u_1, y_1), ..., (u_n, y_n)$ als Beispiele. Das neuronale Netz approximiert die unbekannte, unter Umständen dynamische Abbildung zwischen Ein- und Ausgabe.

Die Parameteradaption kann dabei auf unterschiedliche Arten vorgenommen werden. Wird nach jedem Meßpunkt eine Adaption durchgeführt, so spricht man von *Einzelschrittverarbeitung* . Beim *Batch-Lernen* hingegen wird der Fehler über der gesamten Datenmenge festgestellt und eine entsprechende (Gesamt-)Adaption durchgeführt. Die Einzelschrittverarbeitung wird in der Literatur auch *Online-Lernen* genannt. D. Saad beschäftigt sich in [62] ausführlich in Beiträgen verschiedener Autoren mit dem Thema Online-Lernen in neuronalen Netzen. Die Einzelschrittverarbeitung ist notwendig, wenn die Trainingsdaten zu Beginn des Lernens noch nicht vollständig zur Verfügung stehen [13]. Dies ist beispielsweise der Fall, wenn eine Systemidentifikation on-line durchgeführt wird und die Trainingsdaten erst während des Betriebs anfallen (vgl. Abschnitt 2.5).

Die Art der Abbildung und somit auch das Lernverfahren hängen dabei immer von der gewählten Modellstruktur ab. Im allgemeinsten Fall enthalten die Modelle einen internen Zustand x und einen Parametersatz p und die Abbildungsfunktion bildet in jedem Zeitschritt in Abhängigkeit der Parameter die Eingabe auf die Ausgabe, sowie den aktuellen Zustand auf den Folgezustand ab.

$$(u_i, x_i) \mapsto (y_i, x_{i+1}) \tag{4.2}$$

Mit Hilfe des Zustands ist das Modell in der Lage, dynamische Prozesse nachzubilden. Ist der Zustand allerdings gänzlich durch die aktuelle Eingabe bestimmt und hat keine Auswirkung auf zukünftige Berechnungen, so können nur statische Prozesse modelliert werden (vgl. dazu auch Abschnitt 2.2.1). Der Zustand kann in diesem Fall vernachlässigt werden und man spricht dann von *zustandsfreien* bzw. statischen Modellen. Dynamische Modelle besitzen einen Zustand, der durch Gedächtniselemente realisiert wird. Die Begriffe Zustand und Gedächtnis werden in dieser Arbeit äquivalent verwendet.

Als Spezialfall dieser allgemeinen Modelldefinition lassen sich differenzierbare Modelle angeben, d.h. jene Modelle, deren Abbildungsfunktion f_p abgeleitet werden kann nach ihren Parametern p. Die Lernverfahren können in diesem Fall mit Hilfe von Jacobi-Matrizen einheitlich angegeben werden. Dieser Ansatz wurde in der Modellbibliothek AMoC (ACON Model Classes[2]) [72] verwirklicht. Die Jacobi-Matrix enthält die ersten partiellen Ableitungen einer Funktion nach ihren Parametern. Mit einer differenzierbaren Fehlerfunktion E wird unter Verwendung der Kettenregel folgender Zusammenhang ausgenutzt ($y = f_p(u)$):

[2]Das Projekt ACON - Adaptive Control wurde vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie gefördert (FKZ 01IN510C8). Der Abschlußbericht [5] enthält die erzielten Forschungsergebnisse. Die Modellbibliothek AMoC entstand im Zuge dieses Projektes am Institut für Informatik, Technische Universität München. Sie implementiert die dargestellte Sichtweise in Form von C++-Klassen und beinhaltet Standardmodelle und Optimierungsverfahren.

$$\frac{\partial(E(f_p(u), y_{Prozeß}))}{\partial p} = \frac{\partial E(y, y_{Prozeß})}{\partial y} \cdot \frac{\partial f_p(u)}{\partial p} \tag{4.3}$$

Somit ergibt sich die Ableitung der Fehlerfunktion als Produkt aus den Jacobi-Matrizen der Fehlerfunktion E und des Modells f_p. Letzteres enthält die Implementierung des eigentlichen Lernverfahrens, aber unabhängig von der verwendeten Fehlerfunktion. Dadurch führt eine Änderung der Fehlerfunktion nicht zwangsläufig zu einer kompletten Reimplementierung des Lernfahrens. Lediglich die entstandene Matrixmultiplikation ist in Kauf zu nehmen. Als Fehlerfunktion wird meist die quadratische Fehlerfunktion verwendet.

$$E(y, y_{Prozeß}) = (y - y_{Prozeß})^2 \tag{4.4}$$

Die Lernverfahren für differenzierbare Modelle sind gradientenbasiert und werden als Gradientenabstiegsverfahren bezeichnet. Die grundlegende Idee, sowie die bekanntesten Vertreter dieser Klasse werden im Folgenden erläutert.

4.2.2 Gradientenabstiegsverfahren

Gradientenabstiegsverfahren basieren auf der ersten Ableitung der Fehlerfunktion nach den trainierbaren Parametern, auch als Gradient bezeichnet. Wird der Fehler über dem Parameterraum aufgetragen, so ergibt sich anschaulich ein „Fehlergebirge“ und der Gradient gibt die Richtung und den Betrag der größten Steigung an. Die Idee ist, dem negativen Gradienten zu folgen und somit ein Minimum der Fehlerfunktion zu erreichen. Der Nachteil dieses Verfahrens ist offensichtlich: es kann sich in einem lokalen Minimum verfangen. Es gibt zahlreiche Modifikationen, die versuchen, dieses Problem zu verringern, darunter das bereits erwähnte Verfahren „Simulated Annealing“.

Backpropagation

Der populärste Vertreter der Gradientenabstiegsverfahren ist der *Backpropagation*-Algorithmus [61]. Der Begriff ist so populär, daß er häufig synomym zu Gradientenabstieg verwendet wird. Bei Backpropagation handelt es sich genaugenommen lediglich um ein besonderes Verfahren, für mehrschichtige, vorwärtsgerichtete neuronale Netze[3] die Ableitungen schnell und kostengünstig zu berechnen. Im sog. *Vorwärtsschritt* wird die Netzausgabe

[3] Multilayer-Perzeptron, vgl. Abschnitt 5.1

berechnet. Im sog. *Rückpropagationsschritt* wird der Fehler durch das Netz zurückpropagiert, und dabei werden bereits berechnete Teilergebnisse genutzt. Die Adaptionsformel für einen Parameter p lautet

$$p(t+1) = p(t) - \mu \frac{\partial E}{\partial p} \tag{4.5}$$

Der Parameter μ bezeichnet dabei die Lernrate und steuert die Schrittweite des Verfahrens. Zudem bewirkt die Berücksichtigung des Gradientenbetrags eine automatische Schrittweitensteuerung: im Minimum strebt der Gradient gegen Null. In flachen Gebieten ist die Konvergenz daher sehr langsam. Abhilfe schaffen können dabei Verfahren, die sich hauptsächlich auf das Vorzeichen des Gradienten stützen und die Parameteranpassung unabhängig von der tatsächlichen Größenordnung des Gradienten vornehmen. Der bekannteste Vertreter solcher vorzeichenbasierter Verfahren ist der *RPROP*-Algorithmus [55], [54].

4.3 Probleme bei neuronalen Netzen

Im Folgenden werden einige Probleme der Anwendung neuronaler Netze diskutiert.

Black-Box-Modelle Neuronale Netze fallen in die Klasse der Black-Box-Modelle [67]. Bei Black-Box-Modellen ist ein expliziter Zusammenhang zu physikalisch interpretierbaren Größen nicht erkennbar. Die vorhandenen Modellparameter werden ausschließlich mittels gemessener Prozeßdaten bestimmt. Damit sind neuronale Netze immer dann einsetzbar, wenn kein physikalisches Vorwissen vorliegt. Die mangelnde Interpretierbarkeit allerdings mindert die Bereitschaft für den Einsatz neuronaler Netze. Diese Arbeit verfolgt das Ziel, ein trainiertes neuronales Netz interpretieren zu können in dem Sinne, daß es als Werkzeug zur Identifikation von Totzeiten verwendet werden kann. Einen wichtigen Schritt in Richtung interpretierbare neuronale Modelle liefern auch die Dissertationen [8] und [71].

Zuverlässigkeit Neuronale Netze sind universelle Funktionsapproximatoren [29] und damit theoretisch in der Lage jede beliebige nichtlineare Funktion nachzubilden. Allerdings sind dazu auch gute Ausgangsbedingungen notwendig, d.h. gute Daten. Es sollte daher darauf geachtet werden, möglichst repräsentative Daten über das zu modellierende System zu messen. Meßdaten sind immer etwas verrauscht, sei es, daß Meßeinrichtungen an sich Rauschen verursachen oder sonstige tatsächliche Meßfehler auftreten. Gegebenenfalls empfiehlt sich eine vorausgehende Datenfilterung.

Neuronale Netze besitzen die Fähigkeit, zu lernen und auf unbekannte Daten zu generalisieren. Grundsätzlich kann jedoch nicht garantiert werden, daß das neuronale Netz auf jede beliebige Eingabe eine plausible Ausgabe liefert. Um eine absolute Zuverlässigkeit zu gewährleisten, müßte der gesamte Eingaberaum hinreichend dicht abgetastet werden, was nicht praktikabel ist. Es sind daher Verfahren zu entwickeln, die eine Aussage über die Zuverlässigkeit eines neuronalen Netzes erlauben.

Modellauswahl Beim Einsatz neuronaler Netze stellt sich stets die Frage nach der Auswahl eines geeigneten Netzes. Viele verschiedene neuronale Modelle wurden entwickelt, Lernverfahren modifiziert oder neu definiert. Aus dieser Vielfalt heraus gilt es nun zunächst die Modellklasse festzulegen, die die grundsätzlichen Eigenschaften des Modells bestimmt. Beispielsweise ist zu entscheiden, ob das Netz statisch oder dynamisch sein soll. Kapitel 5 beschäftigt sich mit dieser Fragestellung der Modellauswahl. Je nach Modellklasse sind dann noch die modellspezifischen Parameter geeignet zu besetzen. Wie bereits erwähnt, ist die Festlegung der Modellkomplexität, d.h. die Anzahl der Neurone, ein kritischer Punkt, bei dem abzuwägen ist zwischen hinreichender Approximationsgüte und ausreichender Generalisierungsfähigkeit.

4.4 Neuronale Netze in der Regelungstechnik

Der Einzug der kognitiven Verfahren in die Regelungstechnik kann anhand des Beispiels neuronaler Netze wie folgt geschildert werden: Anfang der 90er Jahre entdeckte die Regelungstechnik den Nutzen der neuronalen Netze für sich und glaubte eine Art Wunderwaffe darin gefunden zu haben - ein Boom an Veröffentlichungen zu diesem Thema war die Folge (vgl. dazu den Überblick in [68]). Nach einiger Zeit jedoch entdeckte man, daß neuronale Netze nicht immer und überall sinnvoll einzusetzen sind — dem Boom folgte die Ernüchterung. Heute ist die Haltung der Regelungstechnik gegenüber neuronalen Netzen als gesund konservativ anzusehen. Erst wenn der Einsatz konventioneller Verfahren nicht möglich ist, werden neuronale Netze verwendet bzw. ergänzend eingesetzt. Im Folgenden wird gezeigt, wie neuronale Netze bisher in der Regelungtechnik eingesetzt wurden. Die Ausführungen stützen sich hauptsächlich auf [31]. Auch [49], [46], [85] und [7] geben einen guten Überblick.

Grundsätzlich ist zu unterscheiden, ob das neuronale Netz als den Ingenieur unterstützendes Werkzeug bei der Systemmodellierung und -identifikation eingesetzt wird oder darüberhinaus als Bestandteil des Regelkreises selbst verwendet wird.

4.4.1 Systemmodellierung

Beim Einsatz neuronaler Netze zur Systemmodellierung werden die folgenden zwei Arten unterschieden.

Direkte Modellierung

Abbildung 4.3 zeigt die Methode der direkten Modellierung. Hier wird das neuronale Netz parallel zu dem zu modellierenden System geschaltet. Mit den Stellwerten als Eingabe und der Systemausgabe als Sollwert lernt das neuronale Netz das Verhalten des Systems. Als Fehlersignal dient die Differenz zwischen der Systemausgabe und der Netzausgabe.

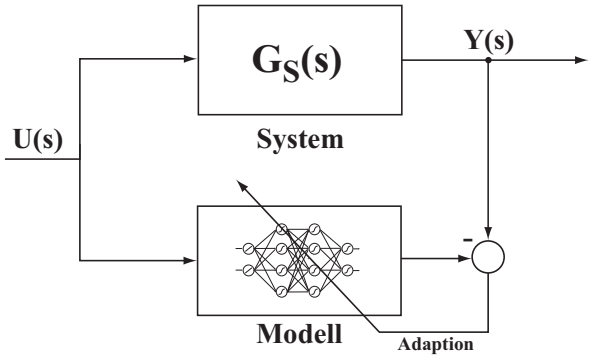

Abbildung 4.3: Direkte Modellierung eines Systems. Das neuronale Netz wird parallel zum System geschaltet. Die Stellwerte werden als Eingabe, die Systemausgabe als Sollwert verwendet. Nach dem Lernen steht das Netz als Modell des Systems zur Verfügung.

Inverse Modellierung

Für den Einsatz bei der Regelung spielt die inverse Modellierung eine wesentlichere Rolle. Abbildung 4.4 zeigt dieses Prinzip. Gegensätzlich zum vorherigen Ansatz wird hier die Systemausgabe dem Netz als Eingabe präsentiert und die Stellwerte dienen als Sollwerte. Das Fehlersignal ist die Differenz zwischen der Netzausgabe und der Systemeingabe. Zu beachten ist hier allerdings, daß die Invertierung nicht eindeutig sein muß.

Beide Modellierungverfahren können dazu verwendet werden, mit Hilfe neuronaler Netze Nichtlinearitäten zu neutralisieren, vgl. Abschnitt 2.2.2. Die Modellierung erfolgt dabei in der Regel off-line. Es wird also eine zeitinvariante Abbildung des vorliegenden Systems gelernt.

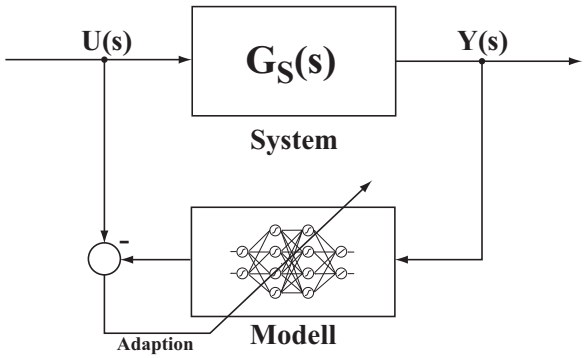

Abbildung 4.4: Inverse Modellierung eines Systems. Im Unterschied zur direkten Modellierung werden bei der inversen Modellierung die Systemausgabe als Eingabe und die Stellwerte als Sollwerte verwendet. Nach dem Lernen ist das Netz ein inverses Modell des Systems.

4.4.2 Regelung

Die in der Systemmodellierung erlangten neuronalen Modelle können auf unterschiedliche Weise als Bestandteil des Regelkreises verwendet werden. Dabei wird unterschieden, ob die Modelle lediglich reproduktiv mitlaufen oder im Betrieb on-line noch lernen sollen. Letzteres ergibt dabei nur dann Sinn, wenn die Komplexität des Netzes und das Lernverfahren ein genügend schnelles Lernen ermöglichen. In beiden Fällen muß gewährleistet sein, daß ein Fehlverhalten des Netzes nicht die Stabilität des Regelkreises gefährdet. Im Folgenden werden die wichtigsten Methoden erläutert.

Regelung mit inversen Modellen

In der Literatur findet sich häufig der Ansatz, ein inverses neuronales Modell direkt als Regler zu benutzen, d.h. das neuronale Netz ersetzt den bisherigen Regler. Diese Methode birgt jedoch einen großen Nachteil: Das neuronale Netz kann auf bisher ungesehene Eingaben mit unsinnigen Stellwerten als Ausgabe reagieren und damit die Stabilität des System gefährden [31].

Sinnvoller ist es, das inverse Modell parallel zum bisherigen Regler einzusetzen und somit eine Vorsteuerung zu erhalten. Das inverse Modell liefert die Stellgröße für das System und der traditionelle Regler fungiert als Korrektur-Mechanismus, um unvorhergesehene Störungen zu eliminieren. Abbildung 4.5 zeigt das Prinzip der Vorsteuerung mit neuronalen Netzen. Abschnitt 8.1 beschreibt ein Beispiel für diese Methode.

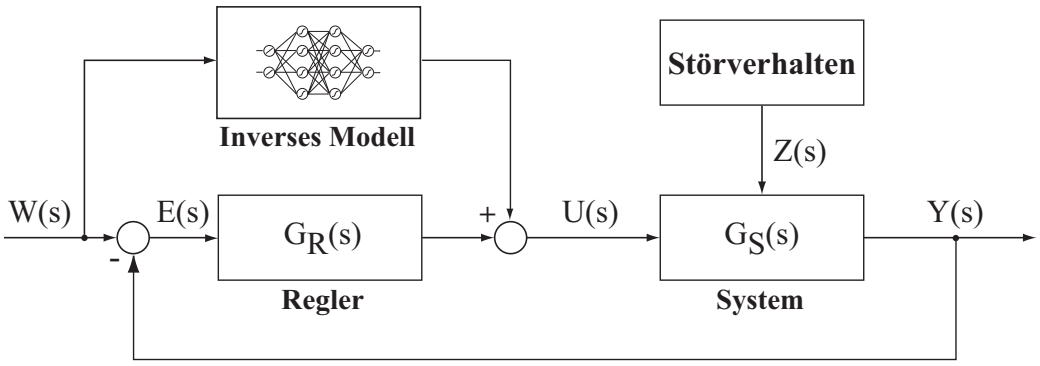

Abbildung 4.5: Regelung durch neuronale Vorsteuerung. Ein inverses Modell des Systems unterstützt einen konventionellen Regler in einem geschlossenen Regelkreis.

Regelung durch Modellvorhersagen

Der Ansatz der Regelung durch Modellvorhersagen ist in der Literatur als *Model Predictive Control (MPC)* bekannt [11, 21]. Bei diesem Verfahren wird ein dynamisches Modell des zugrundeliegenden Systems dazu verwendet, die Systemausgabe für einen festgelegten Zeithorizont vorherzusagen. Als Modell kann dabei ein neuronales Netz eingesetzt werden.

In jedem Regelungsschritt wird eine Kostenfunktion minimiert. Eine typische Kostenfunktion ist zum Beispiel die Summe über die Quadrate der zu erwartenden Regeldifferenz, also der Differenz zwischen der Führungsgröße und der geschätzten Regelgröße. Der Zeithorizont definiert ein Zeitfenster, innerhalb dessen das zukünftige Systemverhalten betrachtet wird. Anhand des Ergebnisses wird ein Regler adaptiert oder aus einer Menge von Reglertypen ausgesucht, was in Bezug auf die Abschätzung des zukünftigen Systemverhaltens optimal geschehen kann.

Eine Kostenfunktion kann nach [11] wie folgt aussehen:

$$\mathrm{J}(N_1, N_2) = E\left[\sum_{i=N_1}^{N_2} \delta(i)(\mathrm{y}(t+i|t) - \mathrm{w}(t+i))^2\right] \tag{4.6}$$

Die Parameter N_1 und N_2 definieren den Zeithorizont. Der Gewichtungfaktor $\delta(i)$ erlaubt eine zeitliche Gewichtung der Zielfunktion. Die Notation $y(t+i|t)$ gibt den zukünftigen Wert der Regelgröße y zum Zeitpunkt $t+i$ an, geschätzt auf der Basis des Wissens zum (aktuellen) Zeitpunkt t.

Liegt beispielsweise ein Totzeitsystem vor, dann kann der Beginn des Zeitfensters N_1 größer

oder gleich der Totzeit gewählt werden, da Regeleingriffe frühestens nach Ablauf dieser Totzeit Auswirkungen haben. Dazu muß allerdings die Totzeit bekannt sein.

Als Nachteile dieser Methode sind zu erwähnen, daß das Modell sehr gut sein muß, um eine genaue Abschätzung der zukünftigen Regelgröße zu geben. Außerdem ist mit diesem Ansatz ein hoher Simulationsaufwand verbunden, da in jedem Regelungsschritt die komplette zukünftige Prozeßtrajektorie simuliert werden muß.

Dieser Abschnitt zeigte einen kurzen Überblick über verschiedene Anwendungsmöglichkeiten neuronaler Netze in der Regelungstechnik. Die vorliegende Arbeit zeigt eine neue Möglichkeit der Verwendung neuronaler Netze auf. Für eine optimale Regelung von Totzeitsystemen ist es wichtig, die Totzeit zu kennen, die dem System innewohnt. Abschnitt 3.4 diskutierte bereits den bisherigen Umgang mit dieser Problematik. In dieser Arbeit werden neuronale Netze dazu verwendet, aus dem Ein-/Ausgabeverhalten eines Systems die darin enthaltene Totzeit zu identifizieren. Das nächste Kapitel enthält die Auswahl eines dafür geeigneten Modells.

Kapitel 5

Modellauswahl

Betrachtet man die Identifikation von Totzeiten in nichtlinearen dynamischen Systemen als Lernproblem für neuronale Netze, so sind spezielle Netzarchitekturen und Lernverfahren notwendig, die das Auftreten der Totzeiten berücksichtigen können. In Abschnitt 4.3 wurde erläutert, daß die Modellauswahl beim Einsatz neuronaler Netze eine wichtige, aber nicht leichte Rolle spielt. In diesem Kapitel wird die Modellauswahl für die neuronale Identifikation von Totzeiten dargelegt.

Nicht betrachtet werden Modelle, die sich durch ihre Nähe zur Biologie auszeichnen und dazu übergehen, einzelne Spikes in den Neuronen zu simulieren. Diese sog. Single-Spike-Netze wären zum Teil in der Lage, Zeitverzögerungen zu lernen, haben jedoch den großen Nachteil eines erheblichen Simulationsaufwands, der aus der Genauigkeit der biologischen Nachbildung resultiert.

5.1 Einfache vorwärtsgerichtete Netze

Vorwärtsgerichtete neuronale Netze sind dadurch charakterisiert, daß die Datenausbreitung in einer Richtung erfolgt, d.h. es gibt keine Rückkopplungen im Netz. Die Neuronen sind in Schichten angeordnet, und jede Schicht ist mit der nachfolgenden Schicht voll vernetzt, d.h. jedes Neuron einer Schicht ist mit jedem Neuron der Folgeschicht verbunden. Die Schichten werden von der Eingabeschicht hin zur Ausgabeschicht aufsteigend durchnummeriert. In einem vorwärtsgerichteten Netz erhält eine Schicht L ihre Eingaben demnach ausschließlich aus Schichten K, für die gilt $K < L$, und gibt ihre Ausgaben ausschließlich an Schichten M, mit $M > L$, weiter.

Die Klassifizierung „einfache vorwärtsgerichtete Netze“ bezeichnet in dieser Arbeit diejenigen vorwärtsgerichteten Netze, die über keinerlei Dynamik verfügen, d.h. zustandsfrei

(statisch) sind. Die Aktivierung der einzelnen Neuronen kann zwar als interner Zustand betrachtet werden, jedoch ist dieser vollständig durch die aktuelle Eingabe bestimmt (vgl. auch S. 31). Die Abschnitte 5.3 und 5.4 hingegen behandeln zustandsbehaftete vorwärtsgerichtete Netze in unterschiedlichen Ausprägungen.

Die Klasse der einfachen vorwärtsgerichteten neuronalen Netze eignet sich sehr gut zur Approximation statischer Funktionen. In [29] wird basierend auf dem Stone-Weierstrass-Theorem bewiesen, daß vorwärtsgerichtete, vielschichtige neuronale Netze die Fähigkeit zur *allgemeinen Funktionsapproximation* besitzen.

Einfache vorwärtsgerichtete neuronale Netze sind derzeit die am weitesten verbreiteten neuronalen Netze und in vielen Anwendungsgebieten erzielen sie gute Ergebnisse [25]. Für die Aufgabenstellung dieser Arbeit jedoch sind sie nicht geeignet. Totzeitsysteme sind zustandsbehaftet (vgl. S. 17), können also nicht mit zustandsfreien Netzen modelliert werden. Insbesondere fehlen einfachen vorwärtsgerichteten neuronalen Netzen Einheiten oder Parameter, die Rückschlüsse auf im zu modellierenden System enthaltene Totzeiten erlauben.

Dem Anwender sind aus dem Bereich der neuronalen Netze hauptsächlich zwei Vertreter aus dieser Klasse bekannt, das *Multilayer-Perzeptron (MLP)* und die *Radial-Basis-Function-Netze (RBF)*. Stellvertetend wird hier nur das Multilayer-Perzeptron kurz vorgestellt, da es sich bei den später vorgestellten dynamischen neuronalen Netzen um Erweiterungen dieses Modells handelt. Bei den RBF-Netzen sei auf die zahlreiche Literatur verwiesen [53, 52, 48, 10].

Multilayer-Perzeptron

Bei diesem Modell handelt es sich um eine Weiterentwicklung von Rosenblatt's Perzeptron [58, 59]. Die Neuronen sind in Schichten angeordnet und diese vollvernetzt. Die erste bzw. letzte Schicht sind die Eingabe- bzw. Ausgabeschicht. Die dazwischenliegenden Schichten werden *versteckte* Schichten genannt (im Englischen: hidden layer). Meist verfügt jede Schicht über einen sogenannten Bias-Vektor $\vec{b}$, der unabhängig von der vorherigen Schicht seinen Wert direkt einbringt. Abbildung 5.1(a) zeigt den Aufbau eines Multilayer-Perzeptrons. Eine alternative Darstellung vollvernetzter neuronaler Netze zeigt Abbildung 5.1(b). Die Neuronen einer Schicht sind zu einem Block zusammengefaßt und ein Pfeil ersetzt die vielen Einzelverbindungen zwischen den Schichten.

Die Eingabe- und die Ausgabeschicht sind linear[1] und die Neuronen der versteckten Schichten verfügen über eine sigmoide Aktivierungsfunktion, die wie folgt definiert ist:

[1] Die Eingabeneuronen dienen lediglich der Aufnahme der Eingabedaten und die Ausgabeneuronen der Bereitstellung der Netzausgabe, vgl. (5.2) auf S. 41

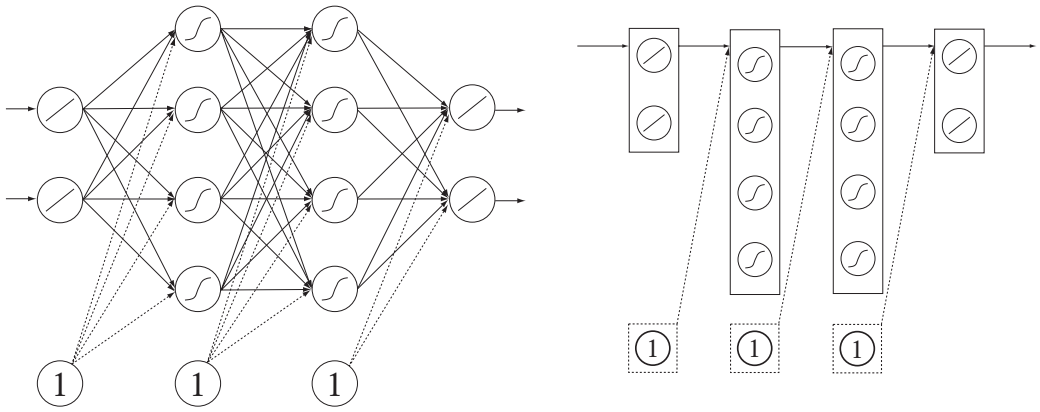

Abbildung 5.1: Aufbau eines Multilayer-Perzeptrons. Abgebildet sind zwei Darstellungen eines vollvernetzten, vierschichtigen Multilayer-Perzeptrons mit je zwei linearen Neuronen in der Eingabe- und in der Ausgabeschicht und jeweils vier sigmoiden Neuronen in zwei versteckten Schichten. Der Bias-Vektor wird über konstante Neuronen eingespeist. (a) zeigt die übliche Darstellung eines MLP mit allen einzelnen Verbindungen zwischen den Neuronen. (b) zeigt eine alternative, vereinfachte Darstellung des gleichen MLP. Die Neuronen einer Schicht sind zu einem Block zusammengefasst. Ein Pfeil zwischen den Blöcken ersetzt die Einzelverbindungen zwischen den Schichten.

$$s : x \mapsto \frac{1}{1 + e^{-x}} \tag{5.1}$$

Die Abbildungsfunktion für Multilayer-Perzeptrone läßt sich rekursiv angeben. Gegeben sei ein MLP mit N Schichten. Bezeichne W_i die Gewichtematrix zwischen zwischen den Schichten $(i-1)$ und i und $\vec{u}$ den Eingabevektor. Dann ergibt sich für die Abbildungsfunktion $f : \vec{u} \mapsto y_N(\vec{u})$:

$$y_i : \vec{u} \mapsto \begin{cases} \vec{u} & i = 0 \\ \vec{b_i} + W_i\, y_{i-1}(\vec{u}) & i = N \\ s(\vec{b_i} + W_i\, y_{i-1}(\vec{u})) & \text{sonst} \end{cases} \tag{5.2}$$

Trainiert wird das Multilayer-Perzeptron mit dem in Abschnitt 4.2.2 bereits beschriebenen Backpropagation-Algorithmus. Dieses numerische Verfahren wurde in den siebziger Jahren von mehreren Forschern unabhängig voneinander entwickelt, aber erst 1986 durch die Arbeit von D. E. Rumelhart und J. L. Mc Clelland [61] populär. Eine ausführliche Behandlung des Algorithmus und seiner Geschichte bietet [57].

5.2 Rekurrente Netze

Im vorherigen Abschnitt wurden Netze vorgestellt, die die Datenausbreitung in nur eine (vorwärtsgerichtete) Richtung erlauben und zudem zustandsfrei sind.

Anders ist dies bei den rekurrenten Netzen, die über rekurrente Verbindungen, d.h. Rückkopplungen im Netz, verfügen. Die aktuelle Aktivierungen der Neuronen hängt damit nicht mehr allein von der aktuellen Eingabe ab, sondern über die rekurrenten Verbindungen auch von früheren Aktivierungen im Netz. Damit besitzen diese Netze einen internen Zustand, der sich auf zukünftige Berechnungen auswirkt. Es gibt keine reine Schichtenstruktur mehr, sondern eine Gewichtematrix W gibt an, ob und wie zwei Neuronen i und j verbunden sind. Ein externer Takt gibt die Berechnungszeitpunkte im Netz vor. Sehr empfindlich reagieren rekurrente Netze bei verrauschten Lerndaten. Durch die Rückkopplungen können sich kleine Abweichungen derart potenzieren, daß es zu völlig sinnlosen Ergebnissen kommt. Die zwei bekanntesten Lernverfahren sind *Backpropagation through time (BPTT)* [83] und *Real Time Recurrent Learning (RTRL)* [87].

Zur Totzeitidentifikation ist dieser Netztyp ungeeignet. Information aus vergangenen Zeitschritten ist nur implizit durch die rekurrenten Verbindungen im Netz vorhanden. Es kann nicht explizit abgelesen werden, welche Informationen aus welcher Zeit das Ergebnis beeinflussen. Bei der Identifikation von Totzeiten soll aber genau dieses herausgefunden werden. In den nächsten Abschnitten werden daher Modelle untersucht, die explizite Verzögerungskomponenten beinhalten.

5.3 Netze mit festen Verzögerungskomponenten

Dieser Abschnitt behandelt neuronale Netze, die das Multilayer-Perzeptron dahingehend erweitern, daß sie Vergangenheitsinformation speichern und somit dynamische Prozesse modellieren können. Die eingefügten Verzögerungskomponenten sind dabei im Gegensatz zum im Abschnitt 5.4 vorgestellten Modell fest, d.h. nicht adaptierbar. Es wird zwischen Netzen mit externer und interner Dynamik unterschieden, je nachdem, ob die Dynamik außerhalb oder innerhalb der Neuronen realisiert wird.

5.3.1 Netze mit externer Dynamik

Bei Netzen mit externer Dynamik werden dem Netz neben den aktuellen Eingaben auch verzögerte Eingaben präsentiert bzw. die Ausgaben der einzelnen Neuronen über mehrere Zeitschritte hinweg gespeichert. Die Netze werden dadurch mit einem Gedächtnis bzw. Zustand versehen.

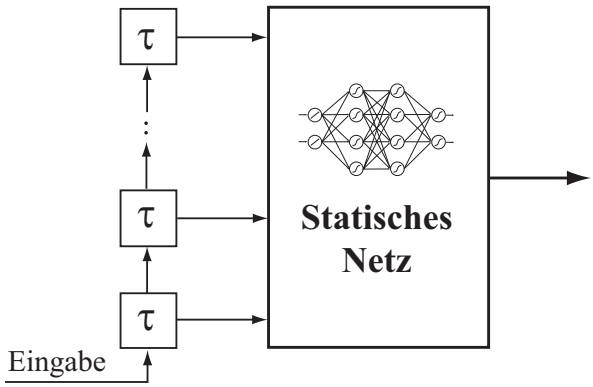

Abbildung 5.2: Aufbau eines einfachen „Time-Delay“-Netzes. Einem statischen Netz sind verzögerte Eingaben vorgeschaltet. Dies können die k letzten Eingaben oder beliebig verzögerte Kombinationen davon sein. Im ersteren Fall wären alle $\tau = 1$ und auch durch das Symbol z^{-1} ersetzbar (vgl. Abschnitt 2.2.3 auf S. 12)

„Time-Delay“-Netze

Wie in Abschnitt 5.1 erläutert, handelt es sich bei einfachen vorwärtsgerichteten neuronalen Netzen um statische Netze. Sie können keine Zeitabhängigkeiten in den Trainingsdaten erkennen. Einem statischen Netz können jedoch neben der aktuellen Eingabe auch in der Vergangenheit liegende Eingaben präsentiert werden. Durch die explizite Verwendung vergangener Meßwerte als zusätzliche Netzeingänge werden die Zeitabhängigkeiten einer dynamischen Abbildung in eine räumliche Ortsabhängigkeit transformiert. Das Gedächtnis ist dabei als eine Art Schieberegister einem statischen Multilayer-Perzeptron vorgeschaltet wie es in Abbildung 5.2 zu sehen ist [15]. Im Wesentlichen wird hier einem statischen Netz Vorwissen mittels Modifikation des Eingaberaums präsentiert.

Die vergangenen Meßwerte können dabei die k letzten Eingaben oder beliebig verzögerte Kombinationen davon sein. Der Vorteil dieses Ansatzes ist, daß der gewöhnliche Backpropagation-Algorithmus verwendet werden kann. Ist Vorwissen vorhanden, so können die verzögerten Werte gezielt ausgewählt werden. Ein unkontrolliertes „Aufblähen“ des Eingabevektors bei mangelndem Vorwissen jedoch führt zu langen Trainingszeiten. Diese Netze können als einfache „Time-Delay“-Netze angesehen werden und wurden beispielsweise in [86] zur Reduktion von Rauschen verwendet.

Einen Schritt weiter gehen die „Time-Delay“-Netze, die in [39, 80] zur Spracherkennung vorgestellt wurden — sie sind unter dem Namen *Time-Delay Neural Network (TDNN)* bekannt. Zusätzlich zur Bereitstellung verzögerter Eingaben werden hier auch in den versteckten Schichten Verzögerungen eingeführt. Abbildung 5.3 zeigt den Aufbau eines TDNN. Aufgrund der fest vorgegebenen Verzögerungswerte kann ein TDNN mit dem Backpropagation-Algorithmus trainiert werden, indem das Netz zeitlich entfaltet wird. Je nach Anzahl der

verwendeten Neuronen und Verzögerungswerte entstehen dadurch jedoch sehr große Netze.

„Time-Delay"-Netze zeigten gute Ergebnisse in der Sprachverarbeitung [39, 80]. Die Netze müssen aber speziell für die jeweilige Aufgabe zugeschnitten werden. Nur mit ausreichend Vorwissen können die Verzögerungswerte sinnvoll gewählt werden. Die Verzögerungen werden fest gesetzt und im Lernverfahren nicht adaptiert. Damit ist das TDNN aber ungeeignet für die Identifikation von Totzeiten, bei der es ja gerade darum geht, meist gänzlich ohne Vorwissen zeitliche Abhängigkeiten in den Trainingsdaten zu lokalisieren.

„Finite-Impulse-Response"-Netze

Bei diesem in [81] vorgestellten Ansatz wird das Multilayer-Perzeptron dahingehend erweitert, daß das einfache Gewicht zur Modellierung einer Synapse ersetzt wird durch einen linearen *Finite-Impulse-Response-Filter (FIR-Filter*[2]*)*. Die Filter ermöglichen wie in (5.3) zu erkennen ist die Speicherung von Vergangenheitsinformation. Die Koeffizienten des Filters können als (Gewichte-)Vektor $W = [w(0), w(1), ..., w(T)]$ dargestellt werden. Die Ausgabe des Filters entspricht der gewichteten Summe $x(k) = \sum_{i=0}^{T} w(i)u(k-i)$ verzögerter Eingaben in ein Neuron, wobei k den (diskreten) Zeitschritt des Netzes angibt. Das Netz ist durch die nachfolgenden Formeln definiert. Abbildung 5.4 zeigt den Aufbau eines FIR-Neurons.

$$\begin{aligned} x_j(k) &= W_{ij}^T U_i(k) \quad \text{mit} \quad U_i(k) = [u_i(k), u_i(k-1), ..., u_i(k-T)] \\ y_j(k) &= f(x_j(k)) \end{aligned} \tag{5.3}$$

FIR-Netze sind äquivalent zu den oben beschriebenen TDNN (siehe dazu [82]). Dies ist leicht ersichtlich, da die eingefügten FIR-Filter nichts anderes tun, als die Ausgaben der vorgeschalteten Neuronen für einige Zeitschritte zu speichern, wie dies auch im TDNN geschieht. Allerdings wurde für die FIR-Netze das Lernverfahren *temporal backpropagation* entwickelt, das eine erweiterte Form des Backpropagation-Algorithmus darstellt und die zeitliche Entfaltung des Netzes erspart [81].

Aufgrund der Äquivalenz zu den „Time-Delay"-Netzen sind die FIR-Netze aus den zuvor genannten Gründen ebenfalls ungeeignet zur Identifikation von Totzeiten.

5.3.2 Netze mit interner Dynamik

Im Gegensatz zu den Netzen des vorherigen Abschnitts werden bei Netzen mit interner Dynamik die Neuronen selbst um einen Zustand erweitert. Im Umfeld der Identifikation dyna-

[2]FIR-Filter sind Filter mit begrenzter Impulsantwortzeit, d.h. ein Impuls endlicher Dauer wirkt sich auch nur in endlicher Zeit auf die Filterausgabe aus (vgl. z.B. [50]).

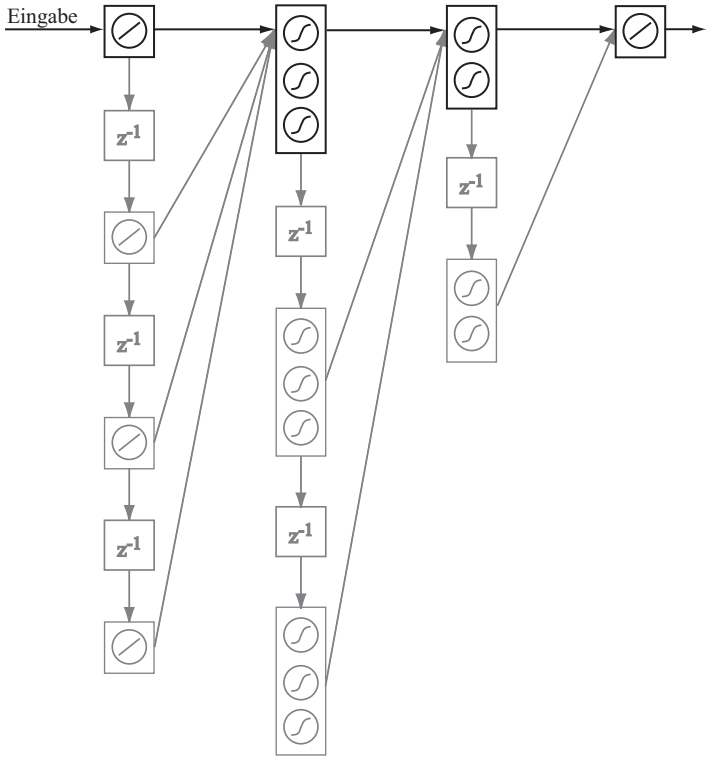

Abbildung 5.3: Aufbau eines TDNN mit Verzögerungselementen in der Eingabeschicht und in den versteckten Schichten. Dargestellt ist ein 4-schichtiges neuronales Netz mit jeweils einem linearen Neuron in der Ein- und in der Ausgabeschicht und mit drei bzw. zwei sigmoiden Neuronen in der ersten bzw. zweiten versteckten Schicht. Die Darstellung des Netzes ist analog zur Abbildung 5.1(b). Die Bezeichnung z^{-1} steht für die Verzögerung um einen Zeitschritt (vgl. Abschnitt 2.2.3 auf S. 12). Die grauen Anteile markieren die Erweiterung gegenüber einem Multilayer-Perzeptron (ohne Bias-Neuronen). Die Ausgaben der Eingabeschicht, der ersten und der zweiten versteckten Schicht werden über drei, zwei bzw. einen Zeitschritt gespeichert und jeweils vollvernetzt in die nachfolgende Schicht gespeist.

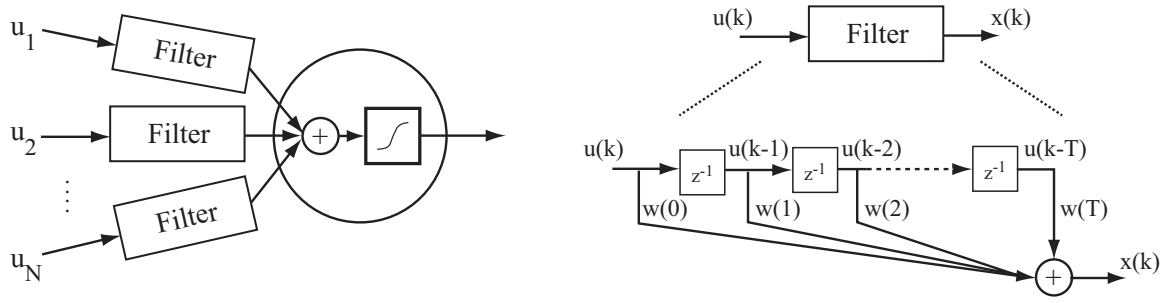

Abbildung 5.4: Aufbau eines Neurons im FIR-Netz. Anstelle eines einzelnen Gewichtes besitzt jede Verbindung einen linearen FIR-Filter. Das ermöglicht im aktuellen Verarbeitungsschritt die Verarbeitung von Informationen aus vergangenen Zeitschritten.

mischer Systeme wurden beispielsweise dynamische Varianten des Multilayer-Perzeptrons (DMLP[3]) und der Radial-Basis-Function-Netze (DRBF[4]) entwickelt. Insbesondere müssen hier keine Vergangenheitswerte am Netzeingang bereit gestellt werden. Beschreibungen der Architektur und der Lernverfahren finden sich in [2] und [3]. Die beiden Netzarten wurden völlig analog dynamisch erweitert, so daß im Folgenden nur das DMLP kurz vorgestellt wird.

Dynamisches MLP

Das dynamische MLP (DMLP) ist ein vorwärtsgerichtetes neuronales Netz mit lokalen Rückkopplungen in den einzelnen Neuronen. Der interne Zustand der Neuronen wird realisiert durch einen linearen Filter 2. Ordnung, der auch als *Auto-Regression-Moving-Average-Filter (ARMA-Filter)* bekannt ist [27].

Die Wahl der Ordnung stellt einen Kompromiß zwischen Flexibilität und Komplexität des Modells dar und bleibt während des Lernverfahrens unverändert. Abbildung 5.5 zeigt den Aufbau eines DMLP-Neurons.

Bezeichne $v(t) = \sum_{i=1}^{N} w_i u_i(t)$ die gewichtete Summation. Die Abbildungsregeln für den internen Zustand $x(t)$ und die Ausgabe $y(t)$ ergeben sich zu

[3]DMLP steht für Dynamic MLP
[4]DRBF steht für Dynamic RBF

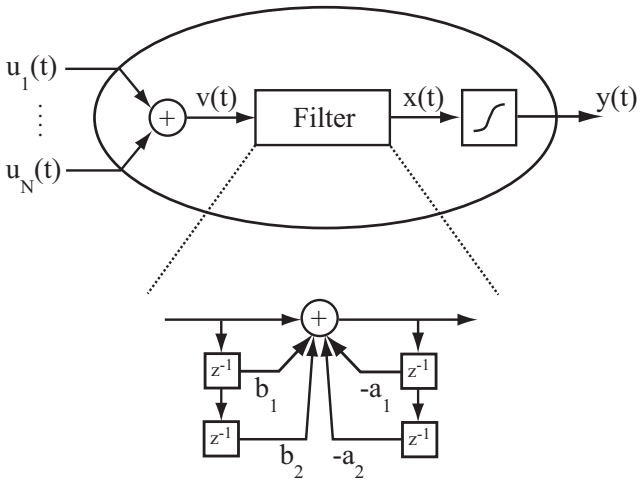

Abbildung 5.5: Aufbau eines Neurons im dynamischen Multilayer-Perzeptron. Das herkömmliche MLP wird durch eine neuron-interne Dynamik erweitert. Diese wird als linearer ARMA-Filter 2. Ordnung realisiert. Das dynamische MLP ist dadurch lokal rekurrent.

$$\begin{aligned} x(t) &= \sum_{k=0}^{2} b_k v(t-k) - \sum_{k=1}^{2} a_k x(t-k) \\ y(t) &= f(x(t)) \end{aligned} \tag{5.4}$$

Unter Verwendung eines Gradientenabstiegverfahrens (dynamische Delta-Regel [2]) werden in diesem Modell neben den Gewichten auch die Filterkoeffizienten a_i, b_i adaptiert.

Diese Modelle mit interner Dynamik eignen sich zwar zur Identifikation dynamischer Prozesse, jedoch nicht zur Identifikation von Totzeiten, da die Verzögerungsordnung ja bereits in der Architektur festgelegt ist.

5.4 Netze mit adaptiven Verzögerungskomponenten

Die Erweiterungen, die im vorherigen Abschnitt vorgestellt wurden, führen an unterschiedlichen Stellen Verzögerungskomponenten ein, die die Verarbeitung von Vergangenheitsinformation in vorwärtsgerichteten neuronalen Netzen erlauben. Allerdings werden diese Verzögerungskomponenten bei der Architekturfestlegung fest gewählt und im Lernverfahren nicht mehr verändert. Zur Identifikation von Verzögerungen sind sie daher nicht geeignet.

Es liegt nahe, die Verzögerungskomponenten adaptiv zu machen. Dazu gab es parallele Entwicklungen. Lin, Dayhoff und Ligomenides stellen in [41] das *Adaptive Time-Delay Neural Network (ATNN)* als Erweiterung der TDNN vor und veröffentlichten in [42, 43] entsprechende Versuchsergebnisse. Sie zeigen die bessere Performanz des Modells bei Zeitreihenvorhersage im Vergleich zum MLP und TDNN. Die Herleitung des Lernalgorithmus in [41] allerdings ist lückenhaft und verwendet schwer nachvollziehbare Berechnungsschritte. Auf Basis der FIR-Netze entwickelten Day und Davenport ungefähr zeitgleich in [16] den gleichen Algorithmus, liefern dafür aber eine bessere Herleitung. In [15] wurde bereits eine kontinuierliche Version des Lernverfahrens „temporal backpropagation“ vorgestellt. Auf dieser Basis wird in [16] der um adaptierbare Verzögerungsparameter erweiterte Algorithmus vorgestellt. In [75] findet sich dessen zeitdiskrete Formulierung. Außerdem wird dort der Algorithmus vereinheitlicht dargestellt, so daß die Modellbibliothek AMoC[5] um das ATNN-Modell erweitert werden konnte.

Das ATNN ist ein mehrschichtiges vorwärtsgerichtetes neuronales Netz. Auf jeder Verbindung ist neben einem Gewicht eine Zeitverzögerung definiert. Das ATNN ist in der Regel schichtenweise vollvernetzt und es können mehrere Verbindungen zwischen zwei Neuronen existieren. Abbildung 5.6 zeigt zwei Neuronen i und j eines ATNN mit n Verbindungen untereinander. Die Gesamtausgabe eines Neurons i zum Zeitpunkt t ergibt sich zu

$$\begin{aligned} y_i(t) &= f(x_i(t)) \\ x_i(t) &= \sum_j \sum_k w_{ijk}(t)\; y_j(t - \tau_{ijk}(t)) \end{aligned} \tag{5.5}$$

wobei $w_{ijk}(t)$ das Gewicht der k-ten Verbindung von Neuron j zu Neuron i und $\tau_{ijk}(t)$ die entsprechende Zeitverzögerung für diese Verbindung zum Zeitpunkt t bezeichnet. Die Verbindungen können als Gedächtnis des Netzes interpretiert werden. Sie verhalten sich wie Puffer, die die Ausgaben des Quellneurons j speichern. Die Zeitverzögerungswerte bestimmen, welche vergangenen Werte aus den Puffern durch das Zielneuron i zur Weiterverarbeitung verwendet werden.

Das Lernverfahren ist ein erweiterter Backpropagation-Algorithmus, in dem neben den Gewichten auch die Zeitverzögerungen adaptiert werden. Der Einsatz dieses Modells beschränkte sich bisher auf Zeitreihenvorhersage [43, 16]. Die Eignung zur Modellierung von Totzeiten in nichtlinearen Systemen wurde erstmals in [75] untersucht. Es wurde zur direkten Modellierung eines Gebäudeheizungssystems [70] verwendet und zeigte im Vergleich zu anderen Modellen (u. a. MLP, TDNN und DMLP) die beste Performanz. Angeregt durch dieses Ergebnis begannen die Arbeiten in Richtung der Identifikation von Totzeiten,

[5]ACON Model Classes: Implementierung diverser neuronaler Modelle in einheitlicher Sichtweise in C++, siehe auch S. 31

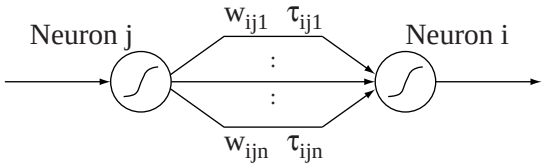

Abbildung 5.6: Zwei Neuronen und deren Verbindung in einem ATNN. Dargestellt sind zwei Neuronen i und j mit n Verbindungen untereinander. Jeder Verbindung werden ein Gewicht w_{ijk} und eine Zeitverzögerung τ_{ijk} zugeordnet.

d.h. die Bestimmung der Totzeit aus dem Ein-/Ausgabe-Verhalten eines Totzeitsystems, welche in diese Arbeit mündeten. Das ATNN hat aufgrund der adaptierbaren Zeitverzögerungsparameter die besten Grundvorraussetzungen zur Totzeitidentifikation. Nachteilig ist jedoch die Tatsache, daß die Zeitverzögerungsparameter ganzzahlig modelliert sind. Das hat die Konsequenz, daß nur ganzzahlige Vielfache der Abtastrate als Totzeit modelliert werden können. Außerdem tritt im Lernverfahren ein Kausalitätsproblem auf, d.h. daß theoretisch in der Zukunft liegende Werte verwendet werden müßten. Da dies offensichtlich nicht möglich ist, wird das Lernverfahren gesteuert über zusätzliche sog. Aging-Parameter verzögert definiert, so daß das Kausalitätsproblem beseitigt wird. Das in [16] vorgeschlagene Lernverfahren ist allerdings sehr starr und beschränkt die Adaption der Zeitverzögerungen erheblich.

Die Identifikation beliebiger Totzeiten in nichtlinearen dynamischen Systemen ist mit diesem Modell nicht möglich. Durch geeignete Erweiterungen kann es jedoch für diese Aufgabe aufbereitet werden. Kapitel 6 stellt die neue Methode zur Identifikation beliebiger Totzeiten in nichtlinearen dynamischen Systemen und zur verbesserten Modellierung und Regelung von Totzeitsystemen vor. Das aus dem ATNN hervorgehende, neue neuronale Netz wird in Abschnitt 6.2 schrittweise entwickelt und in dieser Methode als Kernbaustein verwendet. Der Einsatz neuronaler Netze zur Totzeitidentifikation stellt dabei eine Innovation auf dem Forschungsgebiet der Totzeitidentifikation dar. Die Methode liefert außerdem Bewertungskriterien zur Beurteilung des Identifikationsergebnisses und erlaubt eine einfache Integration von Vorwissen über die zu identifizierende Totzeit. Den erfolgreichen Einsatz dieser neuen Methode belegen Anwendungsbeispiele aus der Mikroelektronik, der Automobilindustrie und der Lebensmitteltechnologie in den Kapiteln 7 und 8.

Kapitel 6

Neuronale Identifikation von Totzeiten

Dieses Kapitel stellt ein neue Methode zur Identifikation von Totzeiten vor. Es beruht im Kern auf einem speziell für diese Aufgabe entwickelten neuronalen Netz. Das neuronale Netz baut auf dem im vorherigen Kapitel vorgestellen Adaptive Time-Delay Neural Network auf (vgl. Abschnitt 5.4) und wird als *EAT-Netz* (*Extended-Adaptive-Timedelay-Netz*) bezeichnet[1].

Aufbauend auf den Arbeiten [75, 76] wird der Lernalgorithmus für das EAT-Netz entwickelt. Es wird definiert, wie aus einem trainierten Netz eine Totzeitidentifikation gewonnen, und wie die Zuverlässigkeit dieser Identifikation beurteilt werden kann. Weiterhin wird dargelegt, wie eine Identifikation beliebiger Totzeiten realisiert wird, bei der unter Umständen vorhandenes Vorwissen eingebracht werden kann.

6.1 Das Gesamtkonzept

In diesem Abschnitt wird zunächst ein Überblick über das Gesamtkonzept der Totzeitidentifikation gegeben, das in Abbildung 6.1 veranschaulicht ist.

Der Ausgangspunkt ist ein Totzeitsystem S_τ mit einer (unbekannten) Totzeit τ. Die Systemeingaben u und die Systemausgaben y werden mit einer bestimmten Abtastrate in einem gewissen Zeitraum meßtechnisch erfaßt, so daß zu diesem System ein (Meß-)Datensatz D_S zur Verfügung steht.

[1] Die hier vorgestellten Arbeiten führten zu den Vorabveröffentlichungen [76], [79] und [77].

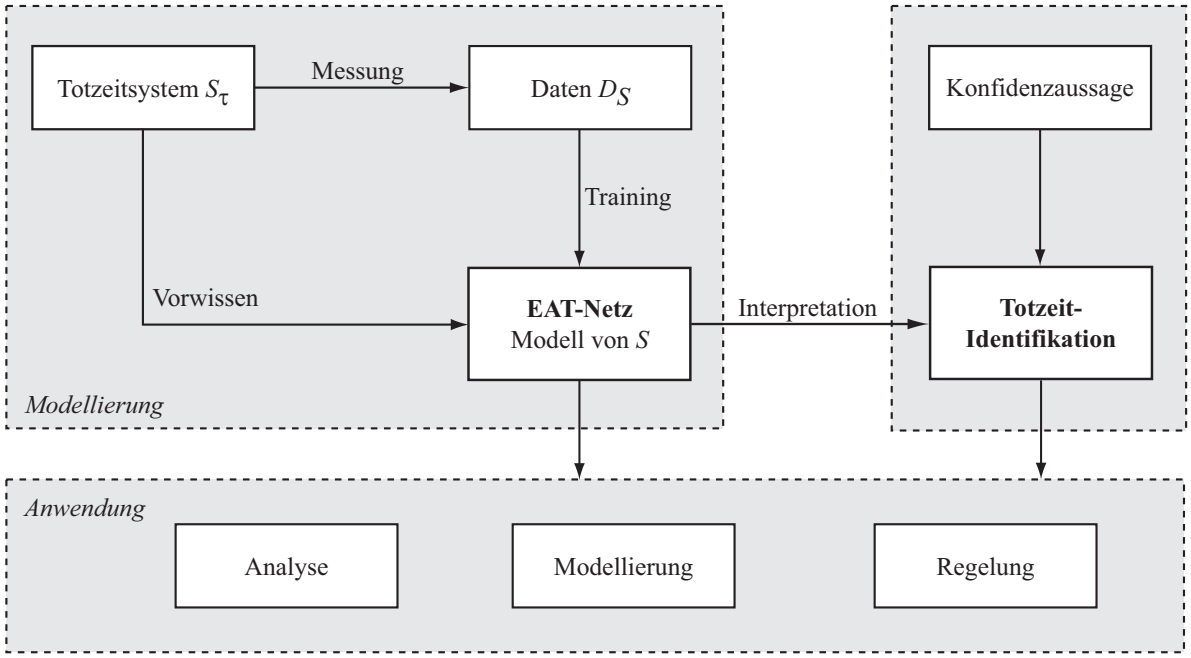

Abbildung 6.1: Neuronale Identifikation von Totzeiten. Gegeben ist ein Totzeitsystem S_τ mit unbekannter Totzeit τ. Anhand von Messungen wird das dynamische neuronale Netz EAT trainiert und stellt nach dem Training ein Modell des Totzeitsystems dar. Durch eine geeignete Interpretation des EAT-Netzes ist die Identifikation der Totzeit möglich, welche durch eine Konfidenzaussage gestützt wird. Sowohl das gewonnene Prozeßmodell, als auch die identifizierte Totzeit können nun zur Analyse, Modellierung oder Regelung innerhalb der gegebenen Anwendung verwendet werden.

Den Kern des Konzepts bildet das EAT-Netz, ein für die Totzeitidentifikation entwickeltes dynamisches neuronales Netz, das in Abschnitt 6.2 vorgestellt wird.

Der erfaßte Datensatz D_S bildet den Trainingsdatensatz für das EAT-Netz. Nach dem Training steht das EAT-Netz als Modell des Totzeitsystems zur Verfügung. Dabei kann unter Umständen vorhandenes Vorwissen über die Totzeit τ in das Modell eingebracht und zur Performanzverbesserung genutzt werden (siehe Abschnitt 6.3.3).

An dieser Stelle kann das EAT-Netz bereits als Prozeßmodell innerhalb eines intelligenten Reglers eingesetzt werden. Kapitel 8 zeigt in Abschnitt 8.1 eine entsprechende Anwendung dieser Möglichkeit.

Darüberhinaus ist durch eine geeignete Interpretation des trainierten neuronalen Netzes die Identifikation der Totzeit τ in S möglich, wie in Abschnitt 6.3.1 dargelegt wird. Wichtig für die Praxistauglichkeit ist die Vertrauenswürdigkeit dieser Identifikation. Anhand einfacher visualisierender und statistischer Mittel wird daher ein Bewertungskriterium für

das Identifikationsergebnis definiert (siehe Abschnitt 6.3.2).

Das Ergebnis einer erfolgreichen Totzeitidentifikation ist ein Totzeitsystem S_τ mit der nun bekannten Totzeit τ. Das neu erworbene Wissen kann in den Reglerentwurf einfließen, so daß das Totzeitsystem S_τ unter Berücksichtigung der Totzeit τ optimal geregelt werden kann. Es kann auch in eine bereits bestehende Modellierung eingebracht werden oder als Grundlage neuer Modellierungstrategien dienen. Anhand des gelieferten Bewertungskriteriums kann der Anwender selbst entscheiden, ob das Ergebnis der neuronalen Identifikation vertrauenswürdig genug ist, um weiterverwendet zu werden.

Damit bildet das in dieser Arbeit entwickelte Konzept zur neuronalen Identifikation von Totzeiten eine wesentliche Grundlage zum Verständnis des Systems einerseits und zur optimalen Systemregelung andererseits.

Besonders hervorzuheben ist die Tatsache, daß die entwickelte neuronale Methode die Identifikation von Totzeiten allein anhand des Ein-/Ausgabeverhaltens des betrachteten Systems ermöglicht. Es bedarf also keiner speziellen Messung direkt vor und hinter dem die Totzeit erzeugenden Übertragungsglied, um die Totzeit identifizieren zu können. Eine derartige spezielle Messung ist in fast allen realen Systemen gar nicht möglich — meßbar sind lediglich die Systemeingaben und -ausgaben. Damit steht mit der neuronalen Identifikationsmethode ein praxistaugliches Werkzeug zur Identifikation von Totzeiten zur Verfügung.

6.2 Das EAT-Netz

6.2.1 Das Basismodell

Das EAT-Netz ist eine Erweiterung des Adaptive Time-Delay Neural Network (vgl. Abschnitt 5.4). Es handelt sich um ein mehrschichtiges, vorwärtsgerichtetes neuronales Netz. Jede Schicht ist mit der nachfolgenden Schicht voll vernetzt, d.h. jedes Neuron einer Schicht ist mit jedem Neuron der Folgeschicht verbunden. Jede Verbindung trägt als adaptierbare Parameter je ein Gewicht w und eine Zeitverzögerung τ. Wie beim ATNN können die Verbindungen als Gedächtnis des Netzes interpretiert werden. Jede Verbindung besitzt ein Quell- und ein Zielneuron und verhält sich wie ein Puffer, der die Ausgaben des Quellneurons speichert. Der Zeitverzögerungswert der Verbindung bestimmt, welcher dieser Ausgabewerte an das Zielneuron zur Weiterverarbeitung weitergeleitet wird. Die Neuronen der Eingabeschicht sind linear und haben lediglich die Aufgabe, die eintreffenden Eingaben in die Puffer der ausgehenden Verbindungen zu verteilen. Ebenso sind die Neuronen der Ausgabeschicht linear. Die Neuronen der versteckten Schichten haben die sigmoide Aktivierungsfunktion, die in (5.1) auf Seite 41 angegeben ist.

Zur Erinnerung wird an dieser Stelle noch einmal die Berechnungsvorschrift eines ATNN-Neurons angegeben.

$$\begin{aligned} y_i(t) &= f(x_i(t)) \\ x_i(t) &= \sum_j \sum_k w_{ijk}(t) \cdot y_j(t - \tau_{ijk}(t)) \end{aligned} \tag{6.1}$$

Die Zeitverzögerungen τ werden im ATNN ganzzahlig modelliert [41]. Wie bereits im vorherigen Kapitel erwähnt, hat dies zur Folge, daß ausschließlich ganzzahlige Vielfache der Abtastrate als ATNN-Zeitverzögerungen modelliert werden können, was eine erhebliche Einschränkung für die Identifikation von Totzeiten bedeutet.

Um beliebige Totzeiten identifizieren zu können, werden die Zeitverzögerungsparameter τ im EAT-Netz als reelle Werte modelliert. Die Verbindungen im EAT-Netz verhalten sich wie Puffer, die die Aktivierungen der Quellneuronen aus vergangenen Zeitschritten speichern. Der Zeitverzögerungswert gibt an, welcher Eintrag aus dem Puffer weiterverarbeitet wird. Um mit einem reellen Zeitverzögerungswert auf einen Puffer zugreifen zu können, wird die Methode der *linearen Interpolation* verwendet. Lineare Interpolation legt die Annahme zugrunde, daß der Zuwachs der Funktionswerte dem Zuwachs der unabhängigen Variablen proportional ist und ist nach [6] wie folgt definiert: liegt der gegebene Wert der unabhängigen Variablen x zwischen den verfügbaren Werten x_0 und $x_1 = x_0 + h$ mit $y_0 = f(x_0)$ und $y_1 = f(x_1) = y_0 + \Delta$, dann kann der funktionale Wert von x als $f(x) = f(x_0) + \frac{x - x_0}{h} \cdot \Delta$ berechnet werden.

Im vorliegenden Fall bilden die Zeitverzögerungsparameter τ die unabhängige Variable einer Funktion κ, die den Zeitverzögerungswert abbildet auf die entsprechend zurückliegende und im Puffer gespeicherte Aktivierung y. Da in jedem Zeitschritt eine Aktivierung gespeichert wird, ergibt sich $h = 1$. τ kann neben der Null jeden beliebigen, positiv reellen Wert annehmen ($\tau \in \mathbb{R}_0$). Für den Fall, daß τ eine natürliche Zahl ist, kann direkt auf den Puffer zugegriffen werden. Andernfalls muß zwischen den beiden verfügbaren, benachbarten Werten interpoliert werden. Die linear interpolierte Aktivierung berechnet sich also wie folgt:

$$\kappa(\tau) \mapsto \begin{cases} y(t - \tau) & \tau \in \mathbb{N}_0 \\ (1 - (\tau - \lfloor \tau \rfloor)) \cdot \kappa(\lfloor \tau \rfloor) + (\tau - \lfloor \tau \rfloor) \cdot \kappa(\lceil \tau \rceil) & \text{sonst} \end{cases} \tag{6.2}$$

Bei der weiteren Definition des Lernalgorithmus wird deutlich werden, daß neben den Aktivierungen y der einzelnen Neuronen, auch andere Vergangenheitsinformation gespeichert

werden muß, beispielsweise werden Gewichtswerte aus vergangenen Zeitschritten benötigt. Daher wird in (6.3) eine allgemeine und um die Zeit t erweiterte Funktionalität der linear interpolierenden Funktion κ angegeben. Dabei bezeichnet ψ die Art der Vergangenheitsinformation.

$$\kappa_\psi : (\tau(t), t) \mapsto \begin{cases} \psi(t - \tau(t)) & \tau(t) \in \mathbb{N}_0 \\ (1 - (\tau(t) - \lfloor \tau(t) \rfloor)) \cdot \kappa_\psi(\lfloor \tau(t) \rfloor) + (\tau(t) - \lfloor \tau(t) \rfloor) \cdot \kappa_\psi(\lceil \tau(t) \rceil) & \text{sonst} \end{cases} \tag{6.3}$$

Mit dieser Interpolationsvorschrift ergibt sich für die Berechnungen eines EAT-Neurons die nachfolgende Vorschrift (vgl. (6.1)).

$$\begin{aligned} x_i(t) &= \sum_j \sum_k w_{ijk}(t) \cdot \kappa_{y_j}(\tau_{ijk}(t)) \\ y_i(t) &= f(x_i(t)) \end{aligned} \tag{6.4}$$

6.2.2 Lernalgorithmus

Der Lernalgorithmus des EAT-Netzes ist eine Erweiterung des Backpropagation-Algorithmus. Herkömmliches Backpropagation berücksichtigt lediglich die Netzgewichte als adaptierbare Parameter. Hier sind sowohl die Gewichte als auch die Zeitverzögerungen adaptierbar. Als Fehlerfunktion wird die folgende quadratische Fehlerfunktion verwendet.

$$E(y, y_{Prozeß}) = \frac{1}{2} \cdot (y - y_{Prozeß})^2 \tag{6.5}$$

Um die Fehlerfunktion E zu minimieren, müssen die Gewichte und die Zeitverzögerungen proportional zum negativen Gradienten angepaßt werden.

$$\begin{aligned} \Delta\tau_{ijk}(t) &= -\eta \cdot \frac{\partial E(t)}{\partial \tau_{ijk}(t)} \\ \Delta w_{ijk}(t) &= -\mu \cdot \frac{\partial E(t)}{\partial w_{ijk}(t)} \end{aligned} \tag{6.6}$$

wobei μ und η die Lernraten darstellen. Nachfolgend werden die Anpassungsregeln für die Gewichte und Zeitverzögerungen hergeleitet. Mit Hilfe der Kettenregel ergibt sich das Produkt aus der Jakobimatrix der Netzfunktion und der Jacobi-Matrix der Fehlerfunktion zu

$$\frac{\partial E(t)}{\partial w_{ijk}(t)} = \frac{\partial E(t)}{\partial y_a(t)} \cdot \frac{\partial y_a(t)}{\partial w_{ijk}(t)} \tag{6.7}$$

wobei $y_a(t)$ die Ausgabe des a-ten Ausgabeneurons bezeichnet. Die Jacobi-Matrix der Fehlerfunktion wird nach

$$\frac{\partial E(t)}{\partial y_a(t)} = (y_a(t) - y_{Ziel_a}(t)) \tag{6.8}$$

berechnet. $y_{Ziel_a}(t)$ bezeichnet dabei den gewünschten Zielwert des Ausgabeneurons a.

Für die Berechnung der Jacobi-Matrix der Netzfunktion wird jedem Neuron i ein Fehlersignal ξ pro Ausgabeneuron a zugeordnet.

$$\xi_i^a(t) := \frac{\partial y_a(t)}{\partial x_i(t)} \tag{6.9}$$

Mit dieser Definition ergibt sich:

$$\begin{aligned} \frac{\partial y_a(t)}{\partial w_{ijk}(t)} &= \xi_i^a(t) \cdot \frac{\partial x_i(t)}{\partial w_{ijk}(t)} \\ &= \xi_i^a(t) \cdot \kappa_{y_j}(\tau_{ijk}(t)) \end{aligned} \tag{6.10}$$

Analog dazu ergibt sich für die Zeitverzögerungswerte

$$\begin{aligned} \frac{\partial y_a(t)}{\partial \tau_{ijk}(t)} &= \xi_i^a(t) \cdot \frac{\partial x_i(t)}{\partial \tau_{ijk}(t)} \\ &= -\xi_i^a(t) \cdot w_{ijk}(t) \cdot \kappa_{\dot{y}_j}(\tau_{ijk}(t)) \end{aligned} \tag{6.11}$$

Die Notation $\dot{y}$ steht für die Ableitung $\frac{dy(t)}{dt}$, die nicht unmittelbar zur Verfügung steht und durch den Differenzenqotienten genähert werden muß (vgl. [75]).

Bei der Berechnung des Fehlersignals ξ ergibt sich eine Fallunterscheidung. Für Ausgabeneuronen kann (6.9) direkt berechnet werden:

$$\xi_i^a(t) = \frac{\partial y_a(t)}{\partial x_i(t)} = \begin{cases} f_i'(x_i(t)) & \text{falls } a = i \\ 0 & \text{sonst} \end{cases} \tag{6.12}$$

Für versteckte Neuronen errechnet sich das Fehlersignal rekursiv aus den Fehlersignalen der jeweils nachgeschalteten Neuronen. Dabei müssen die zeitverzögernden Verbindungen berücksichtigt werden.

$$\begin{aligned} \xi_j^a(t) &= \sum_i \sum_k \frac{\partial y_a(t+\tau_{ijk}(t))}{\partial x_i(t+\tau_{ijk}(t))} \cdot \frac{\partial x_i(t+\tau_{ijk}(t))}{\partial x_j(t)} \\ &= f_j'(x_j(t)) \cdot \sum_i \sum_k \xi_i^a(t+\tau_{ijk}(t)) \cdot w_{ijk}(t+\tau_{ijk}(t)) \end{aligned} \tag{6.13}$$

Zusammenfassend ergeben sich die Anpassungvorschriften für die Gewichte und Zeitverzögerungen zu

$$\Delta\tau_{ijk}(t) = \eta \cdot (y_a(t) - y_{Ziel_a}(t)) \cdot \xi_i^a(t) \cdot w_{ijk}(t) \cdot \kappa_{\dot{y}_j}(\tau_{ijk}(t)) \tag{6.14}$$

$$\Delta w_{ijk}(t) = -\mu \cdot (y_a(t) - y_{Ziel_a}(t)) \cdot \xi_i^a(t) \cdot \kappa_{y_j}(\tau_{ijk}(t)) \tag{6.15}$$

6.2.3 Lernalgorithmus mit Aging

Wie man aus (6.13) erkennt, sind die Anpassungvorschriften (6.14) und (6.15) kausal inkonsistent, da für deren Berechnung zukünftige Werte benötigt werden. Der Fehlerrückfluß muß daher ebenfalls verzögert stattfinden. Man läßt das Lernverfahren künstlich „altern“, indem jedem Neuron ein Alter $a_i \in \mathbb{N}_0$ zugewiesen wird, der sog. *Aging-Wert* [2]. Die Ausgabe eines Neurons in den versteckten Schichten beeinflußt den Netzfehler über p verschiedene Pfade, die von diesem Neuron zur Ausgabeschicht führen. Jeder dieser Pfade besitzt eine bestimmte Gesamtverzögerung, die der Summe der Zeitverzögerungen der einzelnen Verbindungen entspricht, die den Pfad bilden. Der Pfad mit der maximalen Gesamtverzögerung gibt an,

[2] Der Ausdruck Aging kommt von dem englischen Begriff „age“ für Alter.

wie lange die Ausgabe eines Neurons maximal verspätet am Netzausgang erscheint. Daher ist der Aging-Wert eines Neurons größer als diese maximale Gesamtverzögerung von diesem Neuron zur Ausgangsschicht zu wählen. Je weiter entfernt ein Neuron von der Ausgangsschicht ist, desto höher ist sein Aging-Wert. Neuronen der Ausgabeschicht haben den Aging-Wert Null, da hier keine Kausalitätsprobleme gegeben sind (vgl. (6.12)).

Der verzögerte Gradientenabstieg definiert sich zu

$$\frac{\partial E(t)}{\partial w_{ijk}(t-a_i)} = (y_a(t) - y_{Ziel_a}(t)) \cdot \xi_i^a(t-a_i) \cdot \frac{\partial x_i(t-a_i)}{\partial w_{ijk}(t-a_i)} \tag{6.16}$$

Bei der Berechnung des Fehlersignals bleibt die Berechnungsvorschrift für Ausgabeneuronen (6.12) gleich, da der Aging-Wert der Ausgabeschicht gleich Null ist. Die Berechnungsvorschrift für Neuronen in den versteckten Schichten (6.13) ändert sich zu

$$\xi_j^a(t-a_j) = f_j'(x_j(t-a_j)) \cdot \sum_i \sum_k \kappa_{\xi_i^a}(a_j - \tau_{ijk}(t)) \cdot \kappa_{w_{ijk}}(a_j - \tau_{ijk}(t)) \tag{6.17}$$

Damit ergeben sich für (6.14) und (6.15)

$$\Delta\tau_{ijk}(t) = \eta \cdot (y_a(t) - y_{Ziel_a}(t)) \cdot \xi_i^a(t-a_i) \cdot w_{ijk}(t-a_i) \cdot \kappa_{\dot{y}_j}(\tau_{ijk}(t-a_i) + a_i) \tag{6.18}$$

$$\Delta w_{ijk}(t) = -\mu \cdot (y_a(t) - y_{Ziel_a}(t)) \cdot \xi_i^a(t-a_i) \cdot \kappa_{y_j}(\tau_{ijk}(t-a_i) + a_i) \tag{6.19}$$

Die Aging-Parameter sind notwendig, um das Netz im kausal korrekten Zusammenhang zu halten. Die Adaptierungsvorschriften in (6.18) und (6.19) hängen nicht mehr von zukünftigen Werten ab. Allerdings erweitert sich die Speicherung von notwendiger Vergangenheitsinformation neben den Ausgabewerten der Neuronen y auch auf die Gewichte w, Zeitverzögerungsparameter τ und Fehlersignale ξ. Entscheidend ist es nun, wie mit der Bestimmung der Aging-Werte verfahren wird.

Beim ATNN wird das nachfolgende Verfahren zur Festsetzung der Aging-Werte verwendet [16]. Jeder Schicht L wird ein Aging-Wert A_L zugewiesen. Bei der Betrachtung zweier benachbarter Schichten L und $L+1$ gilt für deren Aging-Werte: $A_L > A_{L+1}$. Jedes Neuron trägt den Aging-Wert der Schicht, in der es enthalten ist. Einem Neuron i wird daher der Aging-Wert a_i zugeordnet, mit $a_i = A_L$, falls Neuron i der Schicht L angehört. Der Ausgabeschicht wird der Wert Null zugewiesen. Für jede vorangehende Schicht wird der Aging-Wert um ein festes Inkrement ΔA erhöht, so daß $A_L = A_{L+1} + \Delta A$. Die nachfolgende Tabelle verdeutlicht die Wahl der Aging-Werte bei einem ATNN mit 4 Schichten und $\Delta A = 5$.

	Aging-Wert
Eingabeschicht	15
1. versteckte Schicht	10
2. versteckte Schicht	5
Ausgabeschicht	0

Die Aging-Werte werden bei der Initialisierung des Netzes festgelegt und während des gesamten Lernverfahrens nicht verändert. Dabei wird in [16] argumentiert, daß die Aging-Werte je nach den Anforderungen einer spezifischen Anwendung geeignet zu wählen sind. Weiterhin wird darauf hingewiesen, daß jedes a_i so klein wie möglich gewählt werden sollte, um die Speicheranforderungen zu minimieren, die Stabilität beizubehalten und zu gewährleisten, daß das Lernverfahren nicht zu stark verzögert wird. Dieses bisherige Verfahren ist einfach, aber für die Identifikation von Totzeiten unbefriedigend.

Ein großer Nachteil besteht in der Tatsache, daß die Zeitverzögerungswerte keine beliebigen Werte mehr annehmen können. Die nachfolgende Notation verdeutlicht, welche Schranke auferlegt wird. Wird das verzögerte Fehlersignal ξ wie folgt notiert

$$\xi_i^a(t - a_i) = \xi_i^a(t, a_i) \tag{6.20}$$

dann ergibt sich

$$\begin{aligned} \xi_i^a(t + \tau_{ijk}(t) - a_j) &= \xi_i^a(t + \tau_{ijk}(t) - a_j + a_i - a_i) \\ &= \xi_i^a(t - (a_j - a_i - \tau_{ijk}(t)), a_i) \end{aligned} \tag{6.21}$$

Die Zeitverzögerung τ_{ijk} mit Quellneuron j und Zielneuron i ist also begrenzt auf den Wertebereich $[0..(a_j - a_i)]$. Die Wahl der Aging-Werte wirkt sich damit direkt auf die Adaptionsmöglichkeiten der Zeitverzögerungen aus.

Da die Aging-Werte eine obere Schranke für die anpaßbaren Zeitverzögerungen bilden und im Training des Netzes fest bleiben, können bei ungünstiger Wahl der Aging-Werte zwei Probleme auftreten. Im Training des Netzes sollen neben den optimalen Belegungen der Gewichte auch die optimalen Belegungen der Zeitverzögerungen gefunden werden. Sind die Aging-Werte zu klein, so können die optimalen Belegungen nicht erreicht werden. Sind die Aging-Werte andererseits zu groß, besteht ein zu großer Abstand zu den optimalen Belegungen, so daß das Lernverfahren zu stark verzögert wird. Dies kann dazu führen, daß das Netz sich in suboptimalen Belegungen verfängt. Für viele Anwendungen ist es zudem unmöglich, eine obere Schranke für die Zeitverzögerungen anzugeben, weil diese unbekannt sind. Daher wird in dieser Arbeit ein neuer Ansatz entwickelt.

6.2.4 Neues Aging-Verfahren

Wie bereits erwähnt, ist der Aging-Wert eines Neurons so zu wählen, daß er größer der maximalen Gesamtverzögerung von diesem Neuron zur Ausgabeschicht ist. Eine genaue Berechnung der Aging-Werte ist also verbunden mit der Suche nach dem Pfad mit der maximalen Gesamtverzögerung für jedes einzelne Neuron. Je größer das Netz ist, desto komplexer wird die Berechnung. Im Folgenden wird ein Verfahren eingeführt, das einen Kompromiß darstellt zwischen Genauigkeit und Schnelligkeit.

Dazu werden im Folgenden EAT-Netze geeignet formalisiert. Ein EAT-Netz $\mathcal{N}$ wird als eine geordnete Menge von Schichten angesehen und eine Schicht als eine geordnete Menge von Neuronen.

Definition 6.1 (EAT-Netz) *Ein EAT-Netz $\mathcal{N}$ sei definiert als eine geordnete Menge von Schichten $\mathcal{L}_i$, $i \in \{1, ..., N\}$ mit $N = |\mathcal{N}|$, $N \in \mathbb{N}$*

$$\mathcal{N} := (\mathcal{L}_1, ..., \mathcal{L}_N)$$

Die Schichten $\mathcal{L}_i$ des EAT-Netzes $\mathcal{N}$ seien definiert als geordnete Mengen von Neuronen n^i_j, $i \in \{1, ..., N\}$ und $j \in \{1, .., L_i\}$ mit $L_i = |\mathcal{L}_i|$, $L_i \in \mathbb{N}$

$$\mathcal{L}_i := (n^i_1, ..., n^i_{L_i})$$

Zwei Schichten $\mathcal{L}_i$ und $\mathcal{L}_j$ heißen benachbart, wenn gilt $j = i+1$, mit $\mathcal{L}_j$ ist die Folgeschicht von $\mathcal{L}_i$. Jedes Neuron $n^i_{l_1}$ einer Schicht $\mathcal{L}_i$ ist über k Verbindungen zu jedem Neuron $n^j_{l_2}$ der Folgeschicht verbunden. Jede Verbindung $v = (n^i_{l_1}, n^j_{l_2})$ trägt ein Gewicht $w_{n^j_{l_2} n^i_{l_1} k}$ und eine Zeitverzögerung $\tau_{n^j_{l_2} n^i_{l_1} k}$.

Aufbauend auf diese Sicht auf EAT-Netze wird die maximale Verzögerung zwischen zwei benachbarten Schichten definiert.

Definition 6.2 (Maximale Verzögerung zwischen zwei benachbarten Schichten) *Seien $\mathcal{L}_s$ und $\mathcal{L}_{s+1}$ zwei benachbarte Schichten im EAT-Netz $\mathcal{N}$. Dann ist die maximale Verzögerung $\tau_{s\to(s+1)}$ zwischen diesen beiden Schichten definiert als*

$$\tau_{s\to(s+1)} := max\{\tau_{n^{s+1}_i n^s_j k} : \forall i, j \quad \text{mit} \quad n^{s+1}_i \in \mathcal{L}_{s+1} \in \mathcal{N} \quad \text{und} \quad n^s_j \in \mathcal{L}_s \in \mathcal{N}\}$$

Damit wird das nachfolgende Aging-Verfahren definiert. Wie beim bisherigen Verfahren im ATNN wird jeder Schicht ein Aging-Wert zugeordnet, und jedes Neuron trägt den Aging-Wert der Schicht, in der es sich befindet. Für zwei benachbarte Schichten $\mathcal{L}_i$ und $\mathcal{L}_{i+1}$, gelte für deren Aging-Werte nach wie vor $A_i = A_{i+1} + \Delta A$. Als Inkrement ΔA allerdings wird kein fester Wert verwendet, sondern die maximale Verzögerung zwischen den beiden Schichten berechnet. Nach jedem Lernschritt werden die Aging-Werte nach dem folgenden Aging-Algorithmus berechnet.

Definition 6.3 (Aging-Algorithmus) *Bezeichne A_i den Aging-Wert der Schicht $\mathcal{L}_i$ mit $i \in \{1, ..., N\}$. Die Aging-Werte berechnen sich nach folgender Rekursionsvorschrift*

$$A_i \mapsto \begin{cases} 0 & i = N \\ A_{i+1} + \lceil \tau_{i\to(i+1)} \rceil & sonst \end{cases}$$

Jedes Neuron $n_j^i \in \mathcal{L}_i$ trägt den Aging-Wert $a_j^i = A_i$, also den Aging-Wert der Schicht, in der es sich befindet.

Im Vergleich zu dem im ATNN verwendeten Verfahren liefert dieser Algorithmus ein Lernverfahren, das die Aging-Werte nahe der Zeitverzögerungsparameter hält und somit den Fehlerrückfluß fast optimal verzögert. Außerdem werden den Zeitverzögerungsparametern keine oberen Schranken auferlegt. In Kauf zu nehmen sind allerdings der zusätzliche Aufwand zur Berechnung der Aging-Werte.

6.3 Identifikationsmethodik

Nachdem im vorherigen Abschnitt das EAT-Netz definiert wurde, wird nun festgelegt, wie es als Identifikationsnetz für Totzeiten verwendet wird.

6.3.1 Netzinterpretation

Zunächst ist zu definieren, wie das EAT-Netz im Sinne einer Totzeitidentifikation interpretiert werden kann. Dazu wird der Aufbau eines EAT-Netzes betrachtet. Unter Verwendung der Definition (6.1) ist ein EAT-Netz $\mathcal{N}$ ein vorwärtsgerichtetes neuronales Netz mit $N = |\mathcal{N}|$ Schichten $\mathcal{L}_i$ und jede Schicht besteht aus $L_i = |\mathcal{L}_i|$ Neuronen n_j^i. Benachbarte Schichten sind vollvernetzt und zwischen zwei Neuronen gibt es k Verbindungen. Im EAT-Netz existieren somit p verschiedene Pfade vom Eingang zum Ausgang mit

$$p = k^{N-1} \prod_{i=1}^{N} L_i \tag{6.22}$$

Jeder dieser Pfade hat die Länge $l = |\mathcal{N}| - 1$ und trägt eine Gesamtverzögerung, die der Summe der Zeitverzögerungen der einzelnen Verbindungen entspricht, die dem Pfad angehören. Die nachfolgenden Definitionen legen diese Zusammenhänge fest.

Definition 6.4 (Gesamtverzögerung eines Pfades) *Sei $\mathcal{N}$ ein EAT-Netz und bezeichne $\mathcal{P}$ die Menge aller in $\mathcal{N}$ enthaltenen Pfade p_i mit $p = |\mathcal{P}|$*

$$\mathcal{P} := \{p_1, ..., p_p\}$$

Jeder Pfad p_i ist definiert als eine Folge der einzelnen Verbindungen, die diesen Pfad bilden, wobei jeder Pfad aus $|\mathcal{N}| - 1$ Verbindungen besteht:

$$p_i := (v_1^i, ..., v_{N-1}^i)$$

Bezeichne τ_j^i die zur Verbindung v_j^i gehörige Zeitverzögerung. Dann wird die Gesamtverzögerung τ_i eines Pfades p_i definiert als:

$$\tau_i := \sum_{j=1}^{N-1} \tau_j^i$$

Die Schreibweise $p_i[\tau_i]$ bezeichne den $i-ten$ Pfad mit zugehöriger Gesamtverzögerung τ_i.

Dies ermöglicht die nachfolgende Definition.

Definition 6.5 (Maximaler Pfad) *Sei $\mathcal{P}$ die Menge aller Pfade eines EAT-Netzes. Dann bezeichnet $p_{max}[\tau_{max}]$ den Pfad mit der maximalen Gesamtverzögerung:*

$$\tau_{max} := max\{\ \tau_i \mid p_i[\tau_i]\ ,\ \ p_i \in \mathcal{P}\}$$

$p_{max}[\tau_{max}]$ wird als der maximale Pfad bezeichnet.

Betrachten wir nun noch einmal das Gesamtkonzept aus Abbildung 6.1. Nach dem Training bildet das EAT-Netz ein Modell des Totzeitsystems $\mathcal{S}_\tau$. In diesem Modell gibt der maximale Pfad die maximale Verzögerung des Eingangs auf den Ausgang an und kann damit als die

im modellierten Totzeitsystem $\mathcal{S}_\tau$ enthaltene Totzeit τ interpretiert werden. Dabei ist noch ein folgender Aspekt zu berücksichtigen.

Das EAT-Netz wurde anhand von Trainingsdaten trainiert. Die Trainingsdaten erhält man durch Messung relevanter Prozeßgrößen von $\mathcal{S}_\tau$. Die Messung geschieht dabei so, daß der Prozeß in einem bestimmten Zeitraum mit einer bestimmen Rate abgetastet wird, d.h. die Daten stehen nach der Messung mit einer bestimmten Abtastrate T zur Verfügung. Das EAT-Netz seinerseits arbeitet getaktet. Ein Takt im EAT-Netz entspricht dabei einem Bearbeitungsschritt pro angelegter Eingabe, d.h. für jede eintreffende Eingabe wird im Vorwärtsschritt das Netz berechnet und im Rückpropagationsschritt die Parameter angepaßt. Im nächsten Takt geschieht das Gleiche für die nachfolgende Eingabe. Die vom Netz identifizierte Totzeit bezieht sich auf diesen einheitlichen Netztakt, der nicht der Abtastrate der Daten entspricht. Daher muß der aus dem Netz festgestellte Wert für die Totzeit mit der Abtastrate T verrechnet werden, um auf die im Prozeß enthaltene Totzeit zu schließen: $\tau_{Prozeß} = \tau_{Netz} \cdot T$ für eine beliebige Abtastrate T. In Kapitel 7 wird die Auswirkung der Abtastrate auf das Identifikationsergebnis anhand eines Anwendungsbeispiels dargelegt.

6.3.2 Konfidenzfestlegungen

Bei der Entwicklung dieses neuronalen Identifikationsverfahrens ergab sich beim interdisziplinären Austausch stets die zentrale Fragestellung nach der Zuverlässigkeit des neuronalen Identifikationsergebnisses. Daher werden in diesem Abschnitt Bewertungskriterien für das Identifikationsergebnis festgelegt. Dabei wurde als Rahmenbedingung zugrundegelegt, möglichst einfache Konfidenzmaße zu wählen.

Visualisierung des Lernprozesses für die Totzeit

Als Grundlage der Konfidenzaussage wird zunächst die Visualisierung der Totzeitentwicklung während des Trainings verwendet. Dazu wird die identifizierte Totzeit nach jedem Lernzyklus berechnet und der Gesamtverlauf nach dem Training visualisiert. Ist es in der resultierenden Grafik ersichtlich, daß die Totzeitidentifikation nach einer bestimmten Anzahl von Lernschritten einen stabilen Wert erreicht hat, so wird das Ergebnis als vertrauenswürdig eingestuft. Abbildung 6.2 zeigt je ein Beispiel zu einer vertrauenswürdigen und zu einer nicht vertrauenswürdigen Identifikation. Abbildungen dieser Art werden im Folgenden als *Identifikationsplot* bezeichnet.

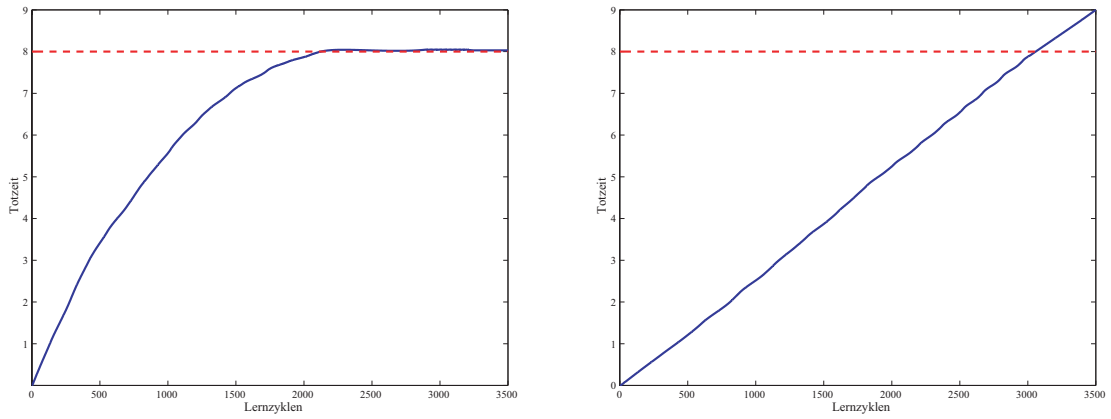

Abbildung 6.2: Visualisierung der Totzeitidentifikation im Identifikationsplot. Die gestrichelte Linie gibt die zu identifizierende Totzeit an. Die durchgezogene Linie stellt die Totzeitentwicklung während des Trainings dar. Links ist ein vertrauenswürdiges Ergebnis zu sehen: die Lernkurve der Totzeit erreicht einen stabilen Wert. Rechts ist ein nicht vertrauenswürdiges Ergebnis abgebildet: der neuronale Identifikationswert konvergiert nicht.

Netzinterne Konfidenzaussagen

Das EAT-Netz liefert zwei netzinterne Konfidenzaussagen. Bei der Interpretation des EAT-Netzes wird die maximale Gesamttotzeit berechnet. Neben dieser Größe kann auch die minimale Gesamttotzeit berechnet werden, die analog zu Definition (6.5) wie folgt festgelegt wird.

Definition 6.6 (Minimaler Pfad) *Sei $\mathcal{P}$ die Menge aller Pfade eines EAT-Netzes. Dann bezeichnet $p_{min}[\tau_{min}]$ den Pfad mit der minimalen Gesamtverzögerung:*

$$\tau_{min} := min\{\ \tau_i \mid p_i[\tau_i]\ ,\ \ p_i \in \mathcal{P}\}$$

$p_{min}[\tau_{min}]$ wird als der minimale Pfad bezeichnet.

Damit erhält man ein Intervall für die identifizierte Totzeit $[\tau_{min}\,;\tau_{max}]$, das im Folgenden als *Identifikationsintervall* bezeichnet wird. Darüberhinaus kann zusätzlich die durchschnittliche Gesamttotzeit im EAT-Netz berechnet werden.

Definition 6.7 (Durchschnittlicher Pfad) *Sei $\mathcal{P}$ die Menge aller Pfade eines EAT-Netzes. Dann bezeichnet $p_{ave}[\tau_{ave}]$ den Pfad mit der durchschnittlichen Gesamtverzögerung:*

$$\tau_{ave} := \frac{\sum_{i=0}^{p} \tau_i}{|\mathcal{P}|} \quad \text{mit} \;\; p_i[\tau_i] \;, \;\; p_i \in \mathcal{P}$$

$p_{ave}[\tau_{ave}]$ wird als der durchschnittliche Pfad bezeichnet.

Die beiden Größen minimale und durchschnittliche Gesamttotzeit können ebenso wie die maximale Gesamttotzeit nach jedem Lernzyklus berechnet und zusammen mit der maximalen Gesamttotzeit im Identifikationsplot visualisiert werden. Mit Hilfe dieser netzinternen Größen kann nun die Güte des Identifikationsergebnisses beurteilt werden. Je enger das Identifikationsintervall ausfällt, desto mehr Vertrauen kann in das Ergebnis gelegt werden.

Zusätzliche statistische Mittel

Die bisherigen Konfidenzaussagen beruhten lediglich auf der Interpretation *eines* EAT-Netzes. Der Aussage eines neuronalen Netz zu vertrauen jedoch stieß im interdisziplinären Austausch auf geringes Vertrauen. Daher wird die Verwendung mehrerer unabhängig voneinander trainierter EAT-Netze empfohlen, um dann über die einfachen statistischen Mittel des Mittelwertes und der Varianz zusätzliche Konfidenzmaße zu erhalten. Die einzelnen EAT-Netze werden nachfolgend als Experten bezeichnet.

Es werden n Experten verwendet, die in ihrer Topologie übereinstimmen und unterschiedliche Initialisierungen besitzen. Bei der Wahl der Topologie geht es darum, wieviele Schichten mit jeweils wievielen Neuronen als Netzstruktur festzulegen sind. Leider gibt es dazu keine allgemeingültige Vorgehensweise. Im Folgenden sollen jedoch einige Hinweise zur Wahl der Topologie gegeben werden. Die Dimension[3] der Eingabe- und der Ausgabeschicht wird durch die jeweilige Anwendung festgelegt und entspricht der Anzahl der zur Verfügung stehenden Ein-/Ausgabegrößen des zugrundeliegenden Systems. Weiterhin ist bekannt, daß vorwärtsgerichtete neuronale Netze mit einer versteckten Schicht universelle Funktionsapproximatoren sind [29]. Daher wird für die EAT-Experten eine dreischichtige Topologie empfohlen. In der Regel reicht eine geringe Anzahl an Neuronen in der versteckten Schicht für eine gute Identifikationsleistung aus. In den Versuchen zu dieser Arbeit (vgl. Kapitel 7) wurden beispielsweise basierend auf experimentellen Voruntersuchungen zehn versteckte Neuronen gewählt. Die Initialisierung eines neuronalen Netzes bestimmt den Startpunkt für die im Training durchgeführte Parameteroptimierung. Durch eine Zufallsinitialisierung der EAT-Experten erhält man verschiedene zufällige Optimierungsstartpunkte. Stellt sich

[3] Unter der Dimension einer Schicht versteht man die Anzahl der Neuronen in dieser Schicht

nach dem Training heraus, daß die EAT-Experten zu einem gemeinsamen Ergebnis konvergieren, so erhöht dies die Vertrauenswürdigkeit der neuronalen Identifikationsmethode, die anhand der nachfolgend definierten Konfidenzmaße überprüft werden kann.

Jeder Experte $\mathcal{N}_i$ liefert eine durchschnittliche Gesamttotzeit $\tau_{ave}^{\mathcal{N}_i}$ und ein Identifikationsintervall $[\tau_{min}^{\mathcal{N}_i}; \tau_{max}^{\mathcal{N}_i}]$. Nach dem Training stehen zu jeder dieser drei Netzgrößen n Werte zur Verfügung, über denen jeweils Mittelwert und Varianz berechnet werden können.

Definition 6.8 (Mittelwerte und Varianzen der Gesamttotzeiten)
Seien $\mathcal{N}_1, ..., \mathcal{N}_n$ n EAT-Experten mit den zugehörigen Größen $\tau_{min}^{\mathcal{N}_i}$, $\tau_{max}^{\mathcal{N}_i}$ und $\tau_{ave}^{\mathcal{N}_i}$. Dann werden die nachfolgenden Mittelwerte und Varianzen für diese Größen definiert.

Maximale Gesamttotzeit:
$$\bar{\tau}_{max} = \frac{1}{n} \sum_i \tau_{max}^{\mathcal{N}_i}$$
$$s_{max}^2 = \frac{1}{n-1} \sum_i (\tau_{max}^{\mathcal{N}_i} - \bar{\tau}_{max})^2$$

Durschnittliche Gesamttotzeit:
$$\bar{\tau}_{ave} = \frac{1}{n} \sum_i \tau_{ave}^{\mathcal{N}_i}$$
$$s_{ave}^2 = \frac{1}{n-1} \sum_i (\tau_{ave}^{\mathcal{N}_i} - \bar{\tau}_{ave})^2$$

Minimale Gesamttotzeit:
$$\bar{\tau}_{min} = \frac{1}{n} \sum_i \tau_{min}^{\mathcal{N}_i}$$
$$s_{min}^2 = \frac{1}{n-1} \sum_i (\tau_{min}^{\mathcal{N}_i} - \bar{\tau}_{min})^2$$

Es kann umso mehr Vertrauen in das Identifikationsergebnis gesetzt werden, je geringer die Ergebnisse der einzelnen Experten schwanken, je kleiner also die oben definierten Varianzen sind. Darüberhinaus können als Identifikationsendergebnis drei Identifikationsintervalle festgelegt werden: je eines für die minimale, die durchschnittliche und die maximale Gesamttotzeit aller Experten. Auch hier gilt, daß das Ergebnis umso besser einzustufen ist, je enger die Intervalle ausfallen.

Definition 6.9 (Identifikationsendergebnis)
Seien $\mathcal{N}_1, ..., \mathcal{N}_n$ n EAT-Experten mit den zugehörigen Größen $\tau_{min}^{\mathcal{N}_i}$, $\tau_{max}^{\mathcal{N}_i}$ und $\tau_{ave}^{\mathcal{N}_i}$. Dann werden als Identifikationsendergebnis die nachfolgenden Identifikationsintervalle für die Gesamttotzeiten festgelegt.

Maximale Gesamttotzeit: $\left[\ min\{\tau_{max}^{\mathcal{N}_i}\}\ ;\ max\{\tau_{max}^{\mathcal{N}_i}\}\ \right]$

Durschnittliche Gesamttotzeit: $\left[\ min\{\tau_{ave}^{\mathcal{N}_i}\}\ ;\ max\{\tau_{ave}^{\mathcal{N}_i}\}\ \right]$

Minimale Gesamttotzeit: $\left[\ min\{\tau_{min}^{\mathcal{N}_i}\}\ ;\ max\{\tau_{min}^{\mathcal{N}_i}\}\ \right]$

6.3.3 Einbringen von Vorwissen

Das entwickelte neuronale Identifikationsverfahren ist datengetrieben, d.h. neuronale Experten werden anhand von Meßdaten des Systems trainiert und daran anschließend als Modell des Systems interpretiert, so daß eine Totzeitidentifikation mit Konfidenzaussage das Ergebnis ist. Zu diesem Verfahren ist prinzipiell gesehen kein Vorwissen notwendig.

Bei Totzeitsystemen $\mathcal{S}_\tau$ liegt in Bezug auf die Systemtotzeit τ allerdings meist Vorwissen in Form eines möglichen Wertebereichs für diese Totzeit vor, das auszunutzen von Vorteil für das Identifikationsergebnis sein kann.

Das neuronale Identifikationsverfahren bietet eine einfache Möglichkeit, dieses Vorwissen einzubringen, indem eine gezielte Vorinitialisierung der Zeitverzögerungsparameter τ vorgenommen wird. Liegt das Vorwissen als Bereichsangabe $\tau \in [\tau_1\,;\tau_2]$ vor, so können die Zeitverzögerungsparameter der EAT-Experten $\mathcal{N}_i$ im nachfolgenden Wertebereich zufällig initialisiert werden:

$$\left[\frac{\tau_1}{N-1}\,;\frac{\tau_2}{N-1}\right]$$

mit $N = |\mathcal{N}_i|$.

In Kapitel 7 wird an einem Anwendungbeispiel gezeigt, daß durch diese gezielte Vorinitialisierung das Identifikationsergebnis verbessert werden kann. Darüberhinaus führt die Vorinitialisierung zu kürzeren Trainingszeiten.

Nachdem in diesem Kapitel das neue neuronale Identifikationsverfahren für beliebige Totzeiten abstrakt-formal beschrieben wurde, folgt im nächsten Kapitel dessen Veranschaulichung an einem Anwendungsbeispiel. Dabei werden auch weiterführende Aspekte wie z.B. Überlegungen zur Abtastrate, zeitvariante Totzeiten und die Identifikation bei verrauschten Daten untersucht.

Kapitel 7

Experimente zur Totzeitidentifikation

Nachdem im vorherigen Kapitel die Vorgehensweise zur Totzeitidentifikation umfassend beschrieben wurde, wird in diesem Kapitel das Funktionieren dieser Methodik anhand von Simulationsergebnissen gezeigt.

Beim Versuchsaufbau handelt es sich um eine Stromregelung bei Zwischenkreisumrichtern, einer speziellen Art von Stromrichtern. Als Stromregelungsverfahren wird das Dead-Beat-Verfahren in den zwei Varianten prädiktiv und nichtprädiktiv verwendet. Das Dead-Beat-Verfahren wurde ausgewählt, da es eine stromfrequenzabhängige Totzeit beinhaltet. Diese Totzeit ist bekannt und kann den Identifikationsergebnissen gegenübergestellt werden. Das Anwendungsbeispiel eignet sich daher zur Überprüfung der in dieser Arbeit vorgestellten neuronalen Identifikation von Totzeiten.

Die Anregung zur Auseinandersetzung mit diesem Anwendungsbeispiel stammt von einem Projektpartner im BMBF-Verbundprojekt ACON[1], dem Lehrstuhl für Elektrische Antriebssysteme von Prof. Dr.-Ing. Dr.-Ing. h.c. D. Schröder an der Technischen Universität München. Die Daten für die Experimente wurden dort mit dem Simulationswerkzeug SABER, einem Verhaltens- und Netzwerksimulator aus dem Bereich der Mikroelektronik, erzeugt. Prof. Schröder und seinen Mitarbeitern sei an dieser Stelle für die Kooperation gedankt. Ein besonderer Dank geht an seine Mitarbeiter Ahmad Radan und Franz Froschhammer für die Zusammenstellung der Daten und für zahlreiche Diskussionstreffen.

Im Folgenden wird zunächst eine Übersicht über Stromrichter gegeben. Daran anschließend werden die beiden Varianten des Dead-Beat-Verfahrens erläutert und schließlich die Identifikationsergebnisse vorgestellt und bewertet.[2]

[1] Nähere Informationen zum Projekt ACON siehe S. 31

[2] Die hier vorgestellten Arbeiten führten zur Vorabveröffentlichung [79].

7.1 Stromrichter

Stromrichter sind Einrichtungen zum Umformen elektrischer Energie [66]. Zwischen zwei energetisch unterschiedlichen Systemen läßt sich mit Hilfe von Stromrichtern der Energiefluß steuern. Bei der Kupplung von Wechsel- und Gleichstromsystemen ergeben sich vier Grundfunktionen der Energieumwandlung. Abbildung 7.1 zeigt die dafür verwendeten Symbole.

Gleichrichter Ein Gleichrichter realisiert die Umformung von Wechselstrom in Gleichstrom.

Wechselrichter Ein Wechselrichter formt Gleichstrom in Wechselstrom um.

Gleichstrom-Umrichter Beim Gleichstrom-Umrichten wird Gleichstrom mit einer gegebenen Spannung und Polarität in Gleichstrom mit einer anderen Spannung und gegebenenfalls der umgekehrten Polarität umgeformt. Es ist auch die Bezeichnung *Gleichspannungswandler* gebräuchlich.

Wechselstrom-Umrichter Hier wird Wechselstrom mit einer gegebenen Spannung, Frequenz und Phasenzahl in Wechselstrom mit einer anderen Spannung, Frequenz und Phasenzahl umgewandelt. Dieser Stromrichtertyp wird auch abgekürzt als *Umrichter* bezeichnet. Umrichter können als Direktumrichter realisiert sein oder als Zwischenkreisumrichter. Letztere bestehen aus einem Gleichrichter und einem Wechselrichter und nehmen eine Wandlung von Wechselstrom (AC[3]) über Gleichstrom (DC[4]) wieder zu Wechselstrom vor und sind die weiter verbreitete Art von Umrichtern.

Aufgabe eines Umrichters ist es, an seinen Ein- und Ausgängen die Spannungen und Ströme auf diejenigen Werte einzustellen, welche durch die Anwendung und den gewünschten Betriebspunkt bestimmt sind. Meist sind dies sinusförmige Verläufe mit bestimmten Amplituden, Phasen und Frequenzen. Die Kurvenformen exakt nachzubilden ist aufgrund der schaltenden Arbeitsweise der Umrichter nicht möglich. Vielmehr ergeben die Schalthandlungen Verzerrungen in den Spannungen und Strömen, welche zum Teil erhebliche Leistungsverluste zur Folge haben. Durch den Einsatz von Regelungsverfahren für Umrichter kann die Verzerrungsblindleistung beeinflußt werden. Es gibt viele verschiedene Regelungsverfahren und oft sind diese für bestimmte Anwendungen optimiert worden. Einen guten Literaturüberblick dazu liefert [33]. Im nächsten Abschnitt wird das nichtprädiktive und das prädiktive Dead-Beat-Regelungsverfahren vorgestellt.

[3] Abkürzung des englischen Begriffs für Wechselstrom: **A**lternating **C**urrent
[4] Abkürzung des englischen Begriffs für Gleichstrom: **D**irect **C**urrent

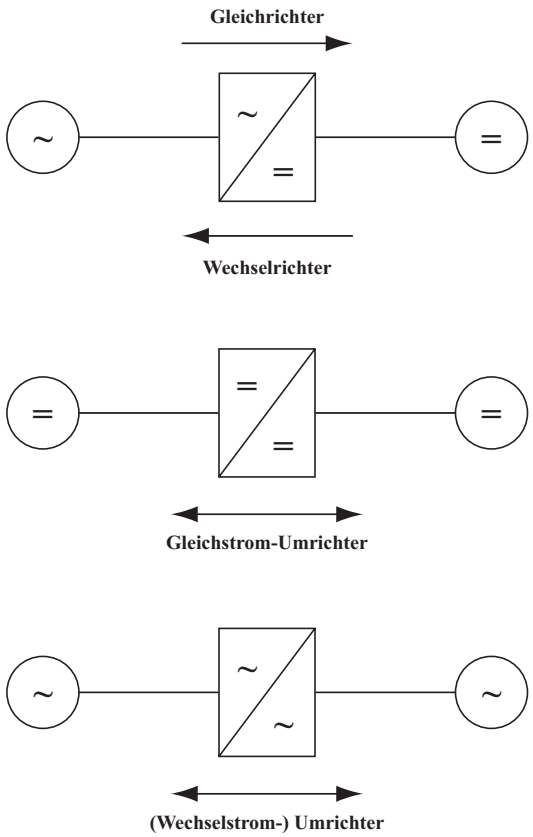

Abbildung 7.1: Grundfunktionen bei Stromrichtern

7.2 Stromregelungsverfahren

Beim Dead-Beat-Verfahren handelt es sich um eine Abtastregelung mit fester Abtastrate T. Es wurde in [23] für Zwischenkreisumrichter (AC/DC/AC-Stromrichter) vorgestellt. Die Bezeichnung des Verfahrens rührt aus der Tatsache, daß das Verfahren in seiner nichtprädiktiven Variante eine Totzeit aufweist, die genau einem Abtastzyklus des Reglers entspricht, und könnte wörtlich mit „Tot-Takt-Verfahren“ übersetzt werden. Die prädiktive Variante hingegen beseitigt diese Totzeit. Beide Varianten werden im stationären Fall betrachtet.

7.2.1 Nichtprädiktives Dead-Beat-Verfahren

Das Dead-Beat-Verfahren ist ein einfaches und effektives Verfahren zur direkten Regelung des Eingangstroms bei AC/DC/AC-Stromrichern. Abbildung 7.2 veranschaulicht den Versuchsaufbau.

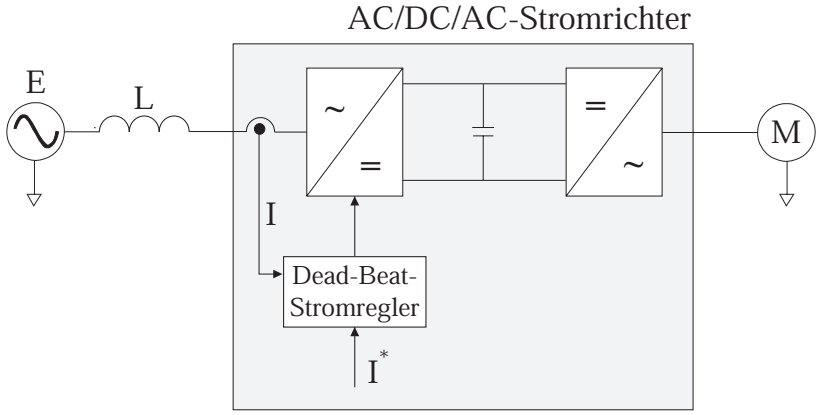

Abbildung 7.2: Veranschaulichung des Versuchaufbaus mit Dead-Beat-Regelung des Eingangsstroms bei Zwischenkreisumrichtern. Diese bestehen aus einem netzseitigen Gleichrichter, der Wechselstrom zu Gleichstrom richtet und einem lastseitigen Wechselrichter, der den Gleichstrom wieder zu Wechselstrom wandelt.

Das Verfahren berechnet mittels einer symmetrischen Pulsweitenmodulation die Referenzspannung U^*. Diese wird benötigt, um den Iststrom I dem Referenzstrom I^* nachzufahren. Dabei erreicht I den Sollwert I^* am Ende eines Abtastintervalls, also nach Ablauf der Totzeit T.

Falls T im Vergleich zur Periode der Gegenspannung E, T_0, als klein angenommen wird, dann kann E als konstant über dem Zeitraum T angenommen werden. Für den n-ten Abtastzyklus, kann die Änderung des Iststroms wie folgt beschrieben werden:

$$\Delta I\Big(nT\Big) = I\Big((n+1)T\Big) - I\,(nT) = \frac{T}{L}\Big(E\,(nT) - U^*\,(nT)\Big) \tag{7.1}$$

Das Regelprinzip fordert, daß der Strom I am Ende des Abtastintervalls den gewünschten Referenzwert I^* erreicht, d.h.

$$I\Big((n+1)T\Big) = I^*(nT) \tag{7.2}$$

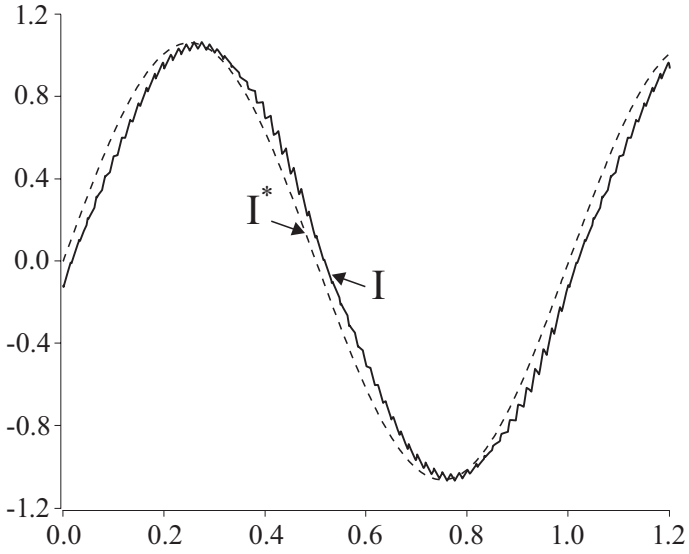

Abbildung 7.3: Verlauf des Iststroms I (durchgezogene Linie) und des Sollstroms I^* (gestrichelte Linie) bei Anwendung der nichtprädiktiven Dead-Beat-Regelung. Der Iststrom ist gegenüber dem Sollstrom um die Totzeit T verschoben.

Durch Einsetzen von (7.2) in Gleichung (7.1) ergibt sich:

$$I^*(nT) - I(nT) = \frac{T}{L}\left(E(nT) - U^*(nT)\right) \tag{7.3}$$

Der Wert der durchschnittlichen Eingangsspannung des Gleichrichters, U^*, während des Abtastintervalls T wird dann wie folgt berechnet:

$$U^*(nT) = E(nT) - \frac{L}{T}\left(I^*(nT) - I(nT)\right) \tag{7.4}$$

Da die Regelung für die Übereinstimmung des Ist- und Sollstroms eine Abtastperiode Zeit benötigt, weisen die Stromverläufe eine Totzeit von T auf, wie deutlich in Abbildung 7.3 zu sehen ist.

7.2.2 Prädiktives Dead-Beat-Verfahren

Die im nichtprädiktiven Dead-Beat-Verfahren vorherrschende Totzeit kann im stationären Fall durch Prädiktion vermieden werden. Anstelle des aktuellen Sollwerts $I^*(kT)$ wird der

geschätzte Wert $I^*((k+1)T)$ des nächsten Intervalls verwendet. Die gemäß Gleichung (7.4) berechnete Sollspannung sorgt dafür, daß der Istwert den gewünschten Sollwert am Ende des Intervalls erreicht. Sind die Sollwerte vorab bekannt, kann die Schätzung durch einfaches Vorziehen des Sollwertes um einen Taktintervall erfolgen. Im stationären Fall ist der zukünftige Verlauf des Sollwertes I^* vorab bekannt, so daß diese Vorgehensweise angewandt werden kann. Die Ergebnisse der Prädiktion sind in Abbildung 7.4 dargestellt.

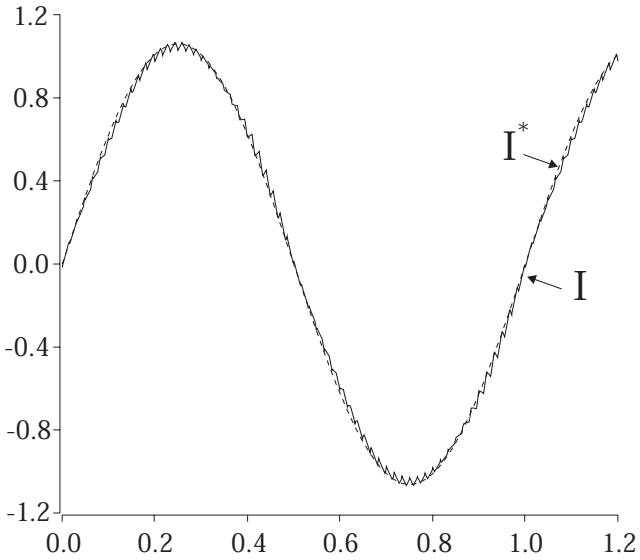

Abbildung 7.4: Verlauf des Iststroms I (durchgezogene Linie) und des Sollstroms I^* (gestrichelte Linie) bei Anwendung der prädiktiven Dead-Beat-Regelung. Die Stromverläufe weisen keine Totzeit auf.

7.3 Identifikationsergebnisse

Im Folgenden werden die Ergebnisse der durchgeführten Versuche beschrieben. Dieser Abschnitt gliedert sich in drei Unterabschnitte. Im ersten Unterabschnitt 7.3.1 werden zunächst einige Festlegungen getroffen, die für alle in diesem Abschnitt aufgeführten Versuche gelten. Unterabschnitt 7.3.2 dokumentiert die Totzeitidentifikation im Falle des nichtprädiktiven Dead-Beat-Verfahrens. Dabei werden die nachfolgenden Aspekte näher untersucht:

- Funktionstüchtigkeit der Totzeitidentifikation
- Identifikation bei Verwendung von Vorwissen

- Identifikation bei verrauschten Daten
- Verwendung unterschiedlicher Abtastraten
- Identifikation zeitvarianter Totzeiten

Der Unterabschnitt 7.3.3 legt die Totzeitidentifikation im Falle des prädiktiven Dead-Beat-Verfahrens dar. Abschnitt 7.4 gibt eine Zusammenfassung und Bewertung aller Ergebnisse. In Abschnitt 7.5 wird eine Anleitung zum systematischen Vorgehen bei der Anwendung der neuronalen Identifikationsmethode gegeben.

7.3.1 Allgemeine Festlegungen

Zur Trainingsdatenerzeugung wurden vier verschiedene Stromfrequenzen F_n angewendet: 25Hz, 50Hz, 75Hz und 100Hz. Demnach standen vier Trainingsdatendateien für die Versuche zur neuronalen Identifikation zur Verfügung. Die Messungen wurden in einem Zeitraum von 0.2 Sekunden erfaßt. Die Dead-Beat-Regelung wurde mit einer Abtastfrequenz F_S gewählt, die abhängig von der Stromfrequenz ist. Die dem Dead-Beat-Verfahren anhängige Totzeit τ_{dbv} errechnet sich in diesem Fall wie folgt:

$$\tau_{dbv} = \frac{1}{F_S} \quad \text{mit} \quad F_S = 72 \cdot F_n \tag{7.5}$$

Entsprechend der angewendeten Stromfrequenzen werden die folgenden vier Versuchsreihen definiert:

- **Versuchsreihe A:** $F_n = 25\text{Hz}$ mit der Totzeit $\tau_{dbv_A} = \frac{1}{72 \cdot 25} = 5,\bar{5} \cdot 10^{-4}$
- **Versuchsreihe B:** $F_n = 50\text{Hz}$ mit der Totzeit $\tau_{dbv_B} = \frac{1}{72 \cdot 50} = 2,\bar{7} \cdot 10^{-4}$
- **Versuchsreihe C:** $F_n = 75\text{Hz}$ mit der Totzeit $\tau_{dbv_C} = \frac{1}{72 \cdot 75} = 1,851 \cdot 10^{-4}$
- **Versuchsreihe D:** $F_n = 100\text{Hz}$ mit der Totzeit $\tau_{dbv_D} = \frac{1}{72 \cdot 100} = 1,3\bar{8} \cdot 10^{-4}$

Da die Zeitverzögerung zwischen dem gewünschten und dem gemessenen Strom auftritt, wird der Referenzwert I^* als Netzeingabe und der Iststrom I als Zielwert benutzt. Die Eingabe- und Zielwerte wurden auf das Intervall [0;1] normiert.

Die verwendeten EAT-Netze sind schichtenweise vollvernetzt mit $k = 1$ für jede Verbindung. Die Anwendung legt die Ein- und Ausgabedimensionen der Netze fest: in diesem

Fall werden ein Eingabeneuron und ein Ausgabeneuron benötigt. Basierend auf Versuchen mit Netzen unterschiedlicher Topologie, erwies sich die Verwendung von zehn Neuronen in einer versteckten Schicht als am günstigsten. Mehr Neuronen bzw. mehr versteckte Schichten trugen nicht zur Verbesserung der Identifikation bei. Die Gewichte werden zufällig im Intervall $[-1; 1]$ und die Zeitverzögerungen (soweit nicht anders vermerkt) auf Null initialisiert.

Für die Beurteilung der Konfidenz der Identifikation werden 10 EAT-Experten verwendet. Um eine möglichst gute Zufallsinitialisierung für jeden EAT-Experten zu erhalten, wird eine einfache Zufallssuche appliziert. Aus jeweils 100 Zufallsinitialisierungen wird die jeweils Beste als EAT-Experte ausgewählt. Die beste Initialisierung ist dabei diejenige mit dem kleinsten Netzfehler nach einmaliger Präsentation der Trainingsdaten. Durch diese einfache Zufallssuche wird für jeden Experten eine günstige Startposition für die im Training durchgeführte Optimierung gewonnen.

Die Identifikationsergebnisse werden in Tabellen zusammengefaßt. Für jede Versuchsreihe wird jeweils das in Definition 6.9 auf S. 66 definierte Identifikationsendergebnis angegeben. Dieses besteht aus den Identifikationsintervallen für die maximale, die durchschnittliche und die minimale Gesamttotzeit. Die Identifikationsintervalle sind mit den entsprechenden Indizes *max*, *ave* und *min* gekennzeichnet. Die Tatsache, daß die Totzeiten bekannt sind, erlaubt die Verifikation der neuronalen Totzeitidentifikation und die Angabe eines prozentualen Fehlers. Zusätzlich werden die Ergebnisse durch ausgewählte Identifikationsplots veranschaulicht. Diese zeigen neben der jeweiligen Zieltotzeit die Entwicklung der maximalen und der minimalen Gesamttotzeit.

7.3.2 Nichtprädiktives Dead-Beat-Verfahren

Dieser Unterabschnitt dokumentiert die im nichtprädiktiven Dead-Beat-Verfahren durchgeführten Versuche.

7.3.2.1 Totzeitidentifikation

Zunächst wird die prinzipielle Funktionstüchtigkeit der in Kapitel 6 entwickelten neuronalen Identifikationsmethode dargelegt. Wie bereits beschrieben werden vier Versuchsreihen verwendet, die sich in der applizierten Stromfrequenz unterscheiden und dadurch unterschiedliche Totzeiten aufweisen. Abbildung 7.5 zeigt exemplarisch einen Ausschnitt des normierten Trainingsdatensatzes der Versuchsreihe B. Es sind die ersten 6000 Datenpunkte von insgesamt 30000 abgebildet.

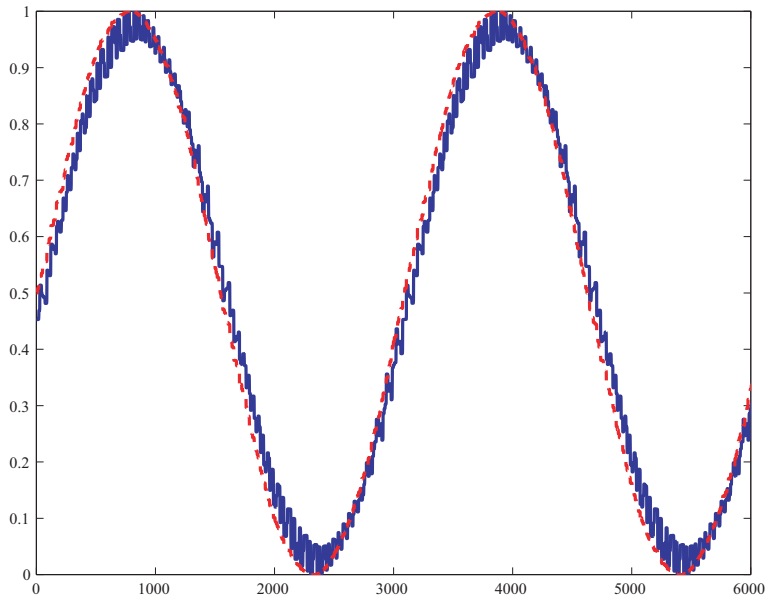

Abbildung 7.5: Trainingsdaten der Versuchsreihe B ($F_n = 50\text{Hz}$) beim nichtprädiktiven Dead-Beat-Verfahren. Zu sehen sind die ersten 6000 Datenpunkte des normierten Trainingsdatensatzes. Die rot-gestrichelte Linie zeigt den Verlauf des Sollstroms, die blau-durchgezogene Linie den Verlauf des Iststroms.

Tabelle 7.1 auf S. 86 faßt die Identifikationsergebnisse aller Versuchsreihen zusammen. Zu erkennen ist eine sehr gute Identifikation der jeweiligen Totzeit. Maßgebend für die Totzeitidentifikation ist die maximale Gesamttotzeit der einzelnen Experten. Die zugehörigen prozentualen Fehler sind in der Tabelle fett markiert. Man erkennt durchweg ein enges Identifikationsintervall mit geringen Abweichungen zur tatsächlich vorliegenden Totzeit. Die netzinternen Konfidenzaussagen, nämlich die Identifikationsintervalle zu den durchschnittlichen und den minimalen Gesamttotzeiten bestätigen das sehr gute Ergebnis. Auch diese Intervalle fallen eng aus und weisen geringe Abweichungen zur Zieltotzeit auf. Insgesamt betrachtet überzeugt das neuronale Identifikationsverfahren, das in dieser Arbeit entwickelt wurde, durch eine nahezu exakte Totzeitidentifikation.

Die Visualisierung der Totzeitentwicklung während des Trainings zeigen ausgewählte Identifikationsplots in Abbildung 7.6 auf S. 87. Deutlich erkennbar ist die hervorragende Konvergenz der Totzeit zum erwünschten Zielwert, sowie eine geringe Breite des Intervalls, das die maximale und die minimale Gesamttotzeit am Trainingsende bilden.

Die nachfolgenden Abschnitte betrachten, wie sich die Totzeitidentifikation unter veränderten Voraussetzungen verhält, beispielsweise bei Berücksichtigung von Vorwissen oder bei Verwendung verrauschter Daten.

7.3.2.2 Einbringen von Vorwissen

Es wird nun untersucht, wie sich das Einbringen von Vorwissen auf die Totzeitidentifikation auswirkt. In diesem Anwendungsbeispiel kann die Tatsache, daß die Totzeiten bekannt sind, als Vorwissen ausgenutzt werden. Normalerweise werden die Zeitverzögerungsparameter in den EAT-Experten auf Null initialisiert. Im Folgenden wird die bekannte Totzeit als obere Intervallgrenze eines möglichen Wertebereichs für die zu identifizierende Totzeit angesehen. Gemäß dem in Abschnitt 6.3.3 vorgeschlagenen Verfahren zur Nutzung vorhandenen Vorwissens über die zu identifizierende Totzeit wird der nachfolgende Wertebereich für die gezielte Initialisierung der Zeitverzögerungsparameter verwendet:

$$\left[0\,;\frac{\tau_{dbv_X}}{2}\right] \quad \text{mit} \quad X \in \{A, B, C, D\} \tag{7.6}$$

Die Division durch 2 ergibt sich aufgrund der dreischichtigen Topologie der verwendeten EAT-Experten (d.h.: $N = 3$). Die Zeitverzögerungsparameter werden in diesem Wertebereich zufällig initialisiert.

Tabelle 7.2 auf S. 88 faßt die Identifikationsergebnisse aller Versuchsreihen zusammen. Wie man deutlich erkennt, haben sich die Identifikationsergebnisse durch das Einbringen von Vorwissen verbessert. Nicht nur die prozentualen Fehler sind deutlich gesunken, sondern auch die Intervallbreite hat sich nochmals verringert. Dies erlaubt die Schlußfolgerung, daß eine gezielte Vorinitialisierung in einem plausiblen Wertebereich dazu beiträgt, die neuronale Identifikation von Totzeiten zu verbessern.

Die Visualisierung der Totzeitentwicklung während des Trainings zeigen ausgewählte Identifikationsplots in Abbildung 7.7 auf S. 89. Dabei wird ein weiterer Vorteil bei Nutzung von Vorwissen sichtbar: der Trainingsaufwand verkürzt sich signifikant. Die Konvergenz gegen die Zieltotzeit wird teilweise mehr als doppelt so schnell erreicht. Besonders deutlich ist dieser Effekt in der Versuchsreihe C zu erkennen.

7.3.2.3 Verrauschte Daten

Wie anfangs dieses Kapitels erwähnt, wurden die für diese Versuche zur Verfügung stehenden Daten mit dem Simulationswerkzeug SABER erzeugt. Es handelt sich also um künstlich generierte Daten. Diese sind exakt in dem Sinne, daß sie keine Meßfehler bzw. kein Meßrauschen beinhalten. Bei Daten, die durch Messungen an realen Systemen hervorgehen ist dies anders. Bei der Vermessung realer Systeme können zahlreiche Störungen auftreten. Beispielsweise können hardwarebedingte Ausreißer oder Signalverluste auftreten, oder die Meßeinrichtungen verursachen Rauschen.

Verrauschte Daten stellen jedoch für die neuronale Identifikation folgendes Problem dar. Neuronale Netze sind datengetrieben, d.h. anhand von Messungen, die dem neuronalen Netz als Eingaben und Ausgaben zur Verfügung stehen, wird das Ziel verfolgt, auf den den Daten zugrundeliegenden Generierungsprozeß zu schließen. Liegen nun aber verrauschte Daten vor, so ist der Rückschluß auf den Generierungsprozeß denkbar schwieriger, da die korrekten Daten durch das Rauschen *verfälscht* sind. Das neuronale Netz muß sozusagen zusätzlich zum eigentlichen Identifikationsaufwand das Rauschen filtern. Ob die neuronale Identifikationsmethode dazu in der Lage ist, wird in diesem Abschnitt untersucht.

Um ein möglichst realistisches Szenario zu schaffen, wurden die zur Verfügung stehenden Daten mit einem simulierten Meßrauschen versehen. Dazu wurde zu allen Signalen weißes Rauschen addiert, dessen maximale Amplitude auf 10% der maximalen Signalamplitude beschränkt war.

In Tabelle 7.3 auf S. 90 sind die Ergebnisse aller Versuchsreihen zusammengestellt. Zunächst ist festzustellen, daß sich die Identifikation im Vergleich zu den Versuchen mit unverrauschten Daten (vgl. Tabelle 7.1) verschlechtert haben. Die Abweichungen von der gewünschten Totzeit sind größer und die Identifikationsintervalle fallen breiter aus. Allerdings liegen die schlechtesten Ergebnisse bei knapp über 6% Abweichung vom tatsächlichen Wert — eine immer noch beachtliche Identifikationsleistung.

Abbildung 7.8 auf S. 91 zeigt einige Identifikationsplots zu diesen Ergebnissen. Zu erkennen sind zunächst Konvergenzschwierigkeiten, die sich in relativ schwankenden Kurvenverläufen und höherem Trainingsaufwand niederschlagen. Deutlich erkennbar ist auch die höhere Intervallbreite bei Trainingsende. Die Gesamttotzeiten erreichen dennoch einen stabilen Wert.

Insgesamt lassen die Ergebnisse die Schlußfolgerung zu, daß das neuronale Identifikationsverfahren auch bei Verwendung verrauschter Daten gut funktioniert.

7.3.2.4 Untersuchungen zur Abtastrate

Ein weiterer interessanter Punkt bei der Identifikation von Totzeiten ist die Auswirkung der Abtastrate auf die Identifikationsgüte.

Die Abtastrate gibt an, in welchem zeitlichen Abstand die Datenpunkte aufeinander folgen, und bestimmt damit die Anzahl der Datenpunkte innerhalb des gegebenen Meßzeitraums. Die zur Verfügung stehenden Trainingsdaten wurden mit drei verschiedenen Raten abgetastet. Es wurden Versuche mit allen Versuchsreihen durchgeführt. An dieser Stelle werden die Ergebnisse exemplarisch anhand der Versuchsreihe C dokumentiert, da sich die Effekte der Abtastrate in dieser Versuchsreihe am besten verdeutlichen lassen. Die Ergebnisse der anderen Versuchsreihen sind dazu vergleichbar. Die applizierten Abtastraten definieren die folgenden Versuchsreihen:

- **Versuchsreihe C1:**
  Abtastrate $T_{C1} = \frac{1}{10^5}$, d.h. ein Datenpunkt alle $1 \cdot 10^{-5}$ Sekunden
- **Versuchsreihe C2:**
  Abtastrate $T_{C2} = \frac{1}{25 \cdot 10^3}$, d.h. ein Datenpunkt alle $4 \cdot 10^{-5}$ Sekunden
- **Versuchsreihe C3:**
  Abtastrate $T_{C3} = \frac{1}{12{,}5 \cdot 10^3}$, d.h. ein Datenpunkt alle $8 \cdot 10^{-5}$ Sekunden

In den ursprünglichen Daten folgen die Datenpunkte alle $6,6 \cdot 10^{-6}$ aufeinander. Durch das Abtasten wurde also die Anzahl der Datenpunkte schrittweise verringert: je höher die Abtastrate, desto weniger Datenpunkte stehen zur Verfügung bzw. desto größer ist der Abstand zwischen zwei aufeinanderfolgenden Datenpunkten.

Tabelle 7.4 auf S. 92 stellt die Ergebnisse dieser Abtast-Versuchsreihen C1, C2 und C3 zusammen. Abbildung 7.9 auf S. 93 zeigt zugehörige Identifikationsplots.

In der Versuchsreihe C1 umfaßt der Trainingsdatensatz 2/3 der Datenpunkte des Trainingsdatensatzes der (ursprünglichen) Versuchsreihe C. Diese Datenreduktion bereitet der neuronalen Identifikationsmethode keine Probleme. Wie man aus Tabelle 7.4 und Tabelle 7.1 entnehmen kann, sind die Abweichungen gegenüber der Zieltotzeit bei beiden Versuchsreihen in ungefähr der gleichen Größenordnung. Auch die Identifikationsplots sind einander sehr ähnlich, bis auf die Tatsache, daß sich die Konvergenz der Gesamttotzeiten bei der Versuchsreihe C1 etwas später einstellt (Abbildung 7.9 oben links) als bei der Versuchsreihe C (Abbildung 7.6 unten links).

In der Versuchsreihe C2 umfaßt der Trainingsdatensatz nur noch 1/6 der Datenpunkte des Trainingsdatensatzes der (ursprünglichen) Versuchsreihe C. Die Identifikationsleistung nimmt deutlich ab. Der prozentuale Fehler steigt signifikant an und die Identifikationsintervalle fallen deutlich breiter aus. Auch beim Identifikationsplot ist eine deutliche Verschlechterung erkennbar (Abbildung 7.9 oben rechts).

In der Versuchsreihe C3 schließlich umfaßt der Trainingsdatensatz 1/12 der Datenpunkte des Trainingsdatensatzes der (ursprünglichen) Versuchsreihe C. Abweichungen von der Zieltotzeit von über 10% sind die Folge. Die Identifikationsintervalle weiten sich aus und die Entwicklung der Gesamttotzeiten (Abbildung 7.9 unten) weisen deutliche Schwankungen auf. Zudem ist keine Konvergenz der Gesamttotzeiten absehbar.

Insgesamt betrachtet läßt sich feststellen, daß eine „Ausdünnung“ der Trainingsdaten zunächst zu keinen signifikanten Veränderungen in der Identifikationsleistung führt. Ein zunehmend grobes Raster in den Trainingsdaten allerdings zieht schlechtere Ergebnisse nach sich. Dies hat vor allem zwei Gründe. Einerseits liegt das Problem bei der neuronalen Identifikationsmethode selbst. Um beliebige Totzeiten modellieren zu können, wurden die Zeitverzögerungsparameter eines EAT-Netzes reellwertig modelliert. Beim Zugriff auf die

Puffer im Netz wird die Methode der linearen Interpolation angewandt. Bei einem zu groben Datenraster sinkt automatisch die Interpolationsgüte und damit die Identifikationsleistung des Netzes. Andererseits liegt hier ein prinzipielles Problem vor. Wenige Datenpunkte zur Verfügung zu haben ist gleichbedeutend damit, wenig Informationen über das System, dem die Daten entnommen sind, vorliegen zu haben. Dadurch verschlechtert sich automatisch die Identifikationsleistung ab dem Moment, zu dem offensichtlich *zu wenig* Daten verwendet werden.

Generell sollte bei der Datenbeschaffung darauf geachtet werden, den zugrundeliegenden Prozeß hinreichend dicht abgetastet zu erfassen. Daher ist eine Verschlechterung der Ergebnisse bei zu wenig zur Verfügung stehenden Daten kein wirklicher Nachteil der neuronalen Identifikationsmethode. Zusammenfassend läßt sich festellen, daß die Identifikation unabhängig von der gewählten Abtastrate funktioniert, ein zu grobmaschiges Abtasten jedoch das Ergebnis verschlechtert.

7.3.2.5 Zeitvariante Totzeiten

Eine Totzeit bzw. ein Totzeitglied ist ein dynamisches System, welches das Eingangssignal um die Zeit τ verzögert und durch die Differenzengleichung $y(t) = u(t-\tau)$ beschrieben werden kann. In den bisherigen Betrachtungen war die Totzeit τ konstant, d.h. zeitinvariant. Es soll nun untersucht werden, wie das neuronale Identifikationsverfahren auf zeitvariante Totzeiten reagiert.

In diesem Anwendungsbeispiel lassen sich leicht Datensätze konstruieren, die eine Änderung der Totzeit beeinhalten. Das Dead-Beat-Verfahren wurde so parametriert, daß sich eine von der applizierten Stromfrequenz abhängige Totzeit ergibt. Indem die einzelnen Trainingsdatensätze aneinandergefügt werden, entsteht ein neuer Datensatz mit einer zeitvarianten Totzeit. Exemplarisch wurden dazu die Versuchsreihen A und B, sowie die Versuchsreihen C und D verknüpft. Dabei wurde darauf geachtet, daß sich der Frequenzwechsel im Nulldurchgang vollzieht. Die Trainingsdaten zu Versuchreihe AB bestehen aus 30000 Datenpunkten. Ungefähr in der Mitte des Datensatzes geht die Stromfrequenz von 25Hz auf 50Hz über. Analog verhält es sich beim Trainingsdatensatz der Versuchsreihe CD.

Das neuronale Identifikationsverfahren liefert ein interessantes Ergebnis, das in Tabelle 7.5 auf S. 94 zusammengefaßt ist. Die maximale Gesamtotzeit konvergiert gegen den maximalen tatsächlichen Totzeitwert (dies ist die Totzeit der Versuchsreihe A bzw. C). Die minimale Gesamttotzeit hingegen konvergiert gegen den minimalen tatsächlichen Totzeitwert (dies ist die Totzeit der Versuchsreihe B bzw. D). Damit liegt als Ergebnis der Wertebereich der zeitvarianten Totzeit vor. Identifikationsplots in Abbildung 7.10 auf S. 94 visualisieren dieses Ergebnis. Ein derartiges Ergebnis wäre im Falle einer konstanten Totzeit aufgrund des großen Abstands zwischen der maximalen und der minimalen Gesamttotzeit

bei Trainingsende auf den ersten Blick wenig vertrauenswürdig. Jedoch konvergieren die Gesamttotzeiten und die Identifikationsintervalle fallen eng aus — ein Hinweis darauf, daß eine zeitvariante Totzeit vorliegt. Je nach Anwendung kann nun entschieden werden, ob die Zeitvarianz der Totzeit berücksichtigt wird oder nicht. Fällt der Wertebereich schmal aus, so wird man meist dazu übergehen, die Zeitvarianz der Totzeit zu vernachlässigen. In diesem Fall wird die obere Intervallsgrenze, somit also die maximale Gesamttotzeit, als Systemtotzeit angenommen. Ist im Vorfeld nicht bekannt, ob die Totzeit zeitinvariant ist oder nicht, so kann dies mit Hilfe des neuronalen Identifikationsverfahrens herausgefunden werden.

7.3.3 Prädiktives Dead-Beat-Verfahren

Die bisherigen Untersuchungen betrachteten die Identifikation der im nichtprädiktiven Dead-Beat-Verfahren vorherrschenden Totzeit. Durch Prädiktion kann diese Totzeit wie in Abschnitt 7.2.2 erläutert vermieden werden. Die mit dem prädiktiven Dead-Beat-Verfahren erzeugten Trainingsdaten weisen daher *keine* Totzeit auf. Abbildung 7.11 auf S. 95 zeigt einen Ausschnitt des normierten Trainingsdatensatzes der Versuchsreihe B im prädiktiven Fall. Man erkennt, daß die Kurvenverläufe des Soll- und des Iststroms keine Verschiebung aufweisen. Das neuronale Identifikationsverfahren darf in diesem Fall keine Totzeit identifizieren.

Es wurden Versuche mit allen vier Versuchsreihen durchgeführt. In jeder Versuchsreihe wurde vom neuronalen Identifikationsverfahren keine Totzeit identifiziert, d.h. die Gesamttotzeiten der EAT-Experten verbleiben bei ihrem Initialwert Null. Tabelle 7.6 auf S. 95 verdeutlicht dies exemplarisch anhand der Versuchsreihe A.

7.4 Bewertung der Ergebnisse

Dieses Kapitel zeigte ein Anwendungsbeispiel aus der Mikroelektronik: die Dead-Beat-Regelung des Eingangstroms bei Zwischenkreisumrichtern. In seiner nichtprädiktiven Variante weist das Verfahren eine Totzeit auf, die zwischen dem Soll- und dem Iststrom auftritt. Das Verfahren wurde so parametriert, daß sich eine stromfrequenzabhängige Totzeit ergab. Vier verschiedene Stromfrequenzen wurden appliziert. Es wurden zahlreiche Aspekte der Totzeitidentifikation untersucht. Zusammenfassend werden folgende Aussagen durch die Versuche belegt:

Beliebige Totzeiten In diesem Anwendungsbeispiel wurden vier Versuchsreihen mit unterschiedlichen, reellwertigen Totzeiten betrachtet. In allen Versuchsreihen gelang

eine sehr gute Identifikation der jeweiligen Totzeit. Damit ist es mit der entwickelten neuronalen Identifikationsmethode möglich, beliebige Totzeiten zu identifizieren.

Zuverlässigkeit Das Identifikationsergebnis muß zuverlässig sein, um weiterverwendet werden zu können. Die eingeführten Bewertungkriterien tragen dazu bei, die Zuverlässigkeit des Ergebnisses zu beurteilen.

Vorwissen Das Ergebnis der Identifikation ohne Nutzung jeglichen Vorwissens über die zu identifizierende Totzeit ist bereits sehr gut. Das Einbringen von Vorwissen durch eine gezielte Initialisierung der Zeitverzögerungsparameter der EAT-Netze ist einfach zu realisieren und trägt zur Verbesserung der Identifikationsergebnisse in mehrfacher Hinsicht bei.

Abtastrate Die neuronale Identifikationsmethode funktioniert prinzipiell unabhängig von der Abtastrate, mit welcher die Daten erfaßt wurden. Ein zu grobes Datenraster jedoch führt zu schlechteren Identifikationsergebnissen, da zu wenige Daten als Informationsträger vorhanden sind.

Verrauschte Daten Die neuronale Identifikationsmethode ist auch bei der Verwendung verrauschter Daten noch erfolgreich.

Zeitvariante Totzeit Bei Daten mit zeitvarianter Totzeit, d.h. die Totzeit ändert ihren Wert innerhalb der zur Verfügung stehenden Daten, liefert die neuronale Identifikationsmethode einen Wertebereich, innerhalb dessen sich die Totzeit bewegt.

Keine Totzeit Die dem nichtprädiktiven Verfahren anhängige Totzeit kann durch Prädiktion vermieden werden. In der prädiktiven Variante des Dead-Beat-Verfahrens liegen also Daten *ohne* Totzeit vor. In diesem Fall liefert die neuronale Identifikationsmethode als identifizierte Totzeit den gewünschten Wert Null.

Insgesamt wurde eine erfolgreiche neuronale Identifikation von Totzeiten dokumentiert, die die gestellten Anforderungen erfüllt, die in der Einleitung dieser Arbeit formuliert wurden (vgl. Abschnitt 1.2). Insbesondere ist es möglich die Totzeit eines nichtlinearen dynamischen Systems allein aus dessen Ein-/Ausgabeverhalten zu identifizieren.

7.5 Methodische Zusammenfassung

Dieser Abschnitt gibt eine methodische Zusammenfassung der neuronalen Identifikationsmethode, die als Anleitung zum systematischen Vorgehen bei deren Anwendung verwendet werden kann.

Den Ausgangspunkt bildet ein Totzeitsystem S_τ, in dem es die Totzeit τ zu identifizieren gilt. Von diesem System werden die meßbaren Eingangs- und Ausgangsgrößen erfaßt. Nach

der Datenerfassung erfolgt die Auswahl und Initialisierung der EAT-Experten. Zunächst ist deren Topologie zu bestimmen. Durch die Anzahl der gemessenen Systemgrößen sind bereits die Dimensionen der Eingabe- und der Ausgabeschicht der EAT-Experten festgelegt. Für die Wahl der versteckten Schichten und Neuronen kann keine allgemeingültige Regel angegeben werden. Durch einige experimentelle Voruntersuchungen kann jedoch schnell eine geeignete Struktur gefunden werden. Empfohlen wird an dieser Stelle die Verwendung einer versteckten Schicht mit einer geringen Anzahl von Neuronen. Die Anzahl der Experten kann beliebig gewählt werden, wobei zu beachten ist, daß mit steigender Expertenanzahl automatisch die Komplexität des Verfahrens ansteigt, insbesondere im Hinblick auf die Auswertung der Ergebnisse. Die Festlegung auf zehn Experten stellt eine sinnvolle Wahl dar.

Bei der Initialisierung der EAT-Experten werden deren trainierbare Parameter mit initialen Werten versehen. Diese bilden den Startpunkt für die im Training durchgeführte Parameteroptimierung. Die Gewichte der EAT-Experten werden zufällig initialisiert. Die Zeitverzögerungsparameter werden abhängig davon initialisiert, ob Vorwissen über die zu identifizierende Totzeit vorliegt oder nicht. Liegt kein Vorwissen vor, so erfolgt die initiale Belegung aller Zeitverzögerungsparameter auf den Wert Null. Ist hingegen Vorwissen vorhanden, so wird die in Abschnitt 6.3.3 vorgeschlagene, gezielte Initialisierung der Zeitverzögerungsparameter verwendet.

Es stehen nun n unterschiedliche EAT-Experten zur Verfügung, die mit den gemessenen Systemdaten belernt werden. Nach dem Training erfolgt die in Abschnitt 6.3.1 definierte Interpretation der EAT-Experten. Als Identifikationsendergebnis stehen dann die in Definition 6.9 festgelegten Identifikationsintervalle zur Verfügung. Dieses Ergebnis kann nun zusammen mit den zugehörigen Identifikationsplots (vgl. Abschnitt 6.3.2) bewertet werden.

Ist in den Identifikationsplots eine gute Konvergenz der Gesamttotzeiten erkennbar und fallen die Identifikationsintervalle eng aus, so liegt ein vertrauenswürdiges Identifikationsergebnis vor. Bei Bedarf können zusätzlich die in Definition 6.8 aufgeführten statistischen Maße zur Bewertung herangezogen werden. Liegen die Grenzwerte der maximalen und der minimalen Gesamttotzeit bei guter Konvergenz und engen Identifikationsintervallen relativ weit auseinander, so deutet dies auf eine zeitvariante Totzeit hin. Das Intervall, das die beiden Gesamttotzeiten nach dem Training bilden, stellt den Wertebereich der zeitvarianten Totzeit dar.

Kein gutes Ergebnis liegt vor, wenn in den Identifikationsplots eine schlechte oder gar keine Konvergenz erkennbar ist oder die Identifikationsintervalle breit ausfallen. Dafür sind mehrere Ursachen denkbar. Zum einen können die Daten zu stark verrauscht sein. Eine Filterung der Daten vor dem Training kann in diesem Fall Abhilfe schaffen. Desweiteren könnte es sein, daß die Daten mit einer unzureichenden Abtastrate erfaßt wurden, so daß nicht genügend Datenpunkte zur Verfügung stehen. Eine erneute Datenerfassung mit einer anderen Abtastrate könnte hier das Problem lösen. Bewirkt die Veränderung der Daten keine Verbesserung der Identifikation, so kann versucht werden, neue EAT-Experten mit

einer veränderten Topologie zu trainieren. Scheitern all diese Verbesserungsversuche, so ist die Totzeit mit der neuronalen Identifikationsmethode nicht identifizierbar.

Nach einer erfolgreich durchgeführten Identifikation kann die identifizierte Totzeit weiterverwendet werden zur Analyse, Modellierung oder Regelung des betrachteten Totzeitsystems.

Neuronale Identifikation			
Versuchsreihe	**Totzeit**	**Ergebnisintervalle**	**Prozentualer Fehler**
A	$5,\bar{5} \cdot 10^{-4}$	$[5,499\,;5,675]_{max}$	**0,89 – 2,16%**
		$[5,409\,;5,616]_{ave}$	0,76 – 2,63%
		$[5,349\,;5,485]_{min}$	1,26 – 3,71%
B	$2,\bar{7} \cdot 10^{-4}$	$[2,761\,;2,822]_{max}$	**0,37 – 1,61%**
		$[2,678\,;2,792]_{ave}$	0,51 – 3,56%
		$[2,671\,;2,758]_{min}$	0,68 – 3,84%
C	$1,851 \cdot 10^{-4}$	$[1,856\,;1,885]_{max}$	**0,26 – 1,78%**
		$[1,803\,;1,873]_{ave}$	0,83 – 2,62%
		$[1,793\,;1,836]_{min}$	0,87 – 3,17%
D	$1,3\bar{8} \cdot 10^{-4}$	$[1,375\,;1,398]_{max}$	**0,28 – 0,96%**
		$[1,350\,;1,396]_{ave}$	0,56 – 2,78%
		$[1,338\,;1,383]_{min}$	0,43 – 3,60%

Tabelle 7.1: Neuronale Identifikation der Dead-Beat-Totzeiten im nichtprädiktiven Fall. Die Tabelle faßt die Identifikationsendergebnisse für alle Versuchsreihen zusammen. Die Identifikationsendergebnisse bestehen aus den Identifikationsintervallen für die maximale, die durchschnittliche und die minimale Gesamttotzeit. Deutlich zu erkennen ist eine nahezu exakte Identifikation der jeweiligen Totzeit bei allen Versuchsreihen. Die Intervalle sind eng gefaßt und liegen nahe aneinander. Die prozentualen Fehler zu den Identifikationsintervallen der maximalen Gesamttotzeit sind fett markiert. Zugunsten der Übersichtlichkeit wurde bei den Identifikationsintervallen der Faktor 10^{-4} weggelassen. Abbildung 7.6 auf S. 87 visualisiert ausgewählte Ergebnisse in Identifikationsplots.

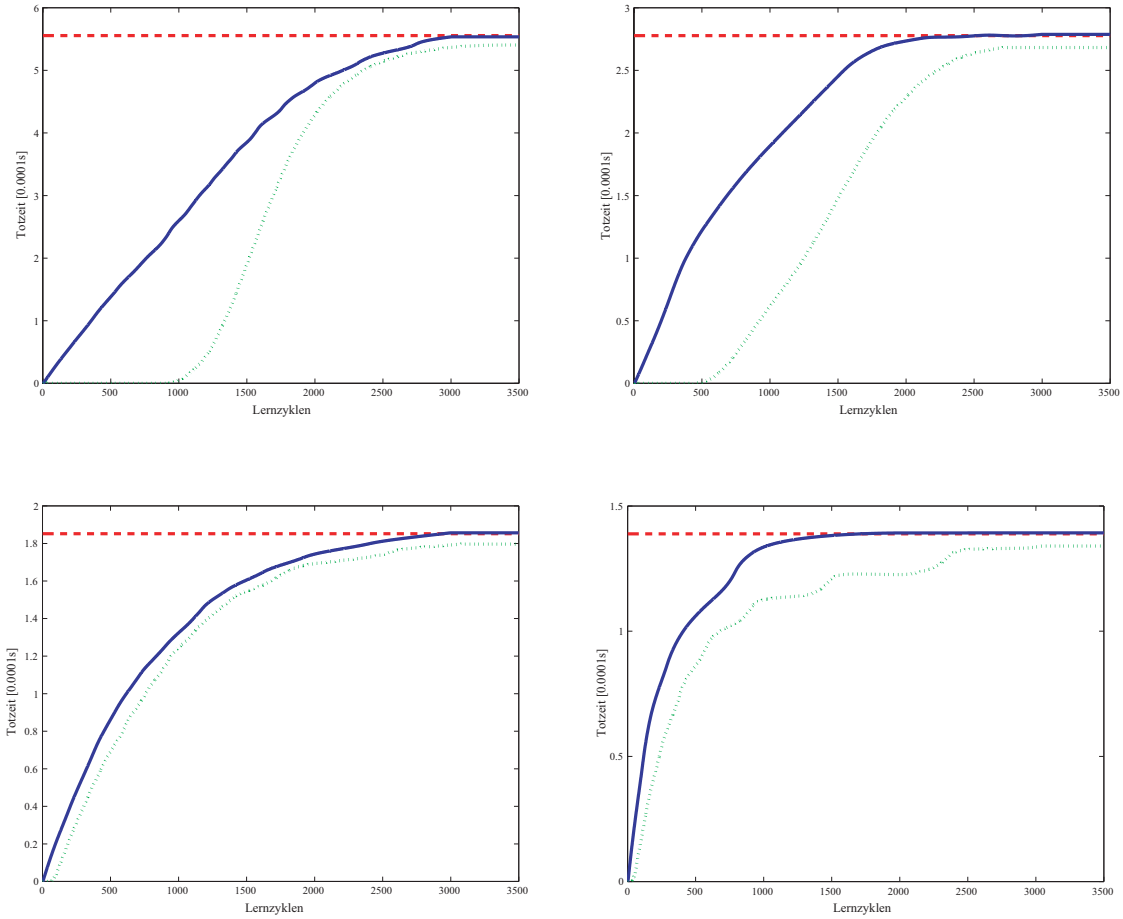

Abbildung 7.6: Identifikationsplots beim nichtprädiktiven Dead-Beat-Verfahren. Zu sehen sind Identifikationsplots zu den einzelnen Versuchsreihen. Oben links: Versuchsreihe A. Oben rechts: Versuchsreihe B. Unten links: Versuchsreihe C. Unten rechts: Versuchsreihe D. Die horizontale, rot-gestrichelte Linie gibt jeweils den Wert der zu identifizierenden Totzeit an. Die Identifikationsplots visualisieren die Entwicklung der maximalen (blau-durchgezogen) und der minimalen (grün-gepunktet) Gesamttotzeit der jeweils besten EAT-Experten (aus Tabelle 7.1). Deutlich zu erkennen sind einerseits die ausgesprochen gute Konvergenz zur jeweiligen Zieltotzeit und andererseits die geringe Breite des Intervalls, das die maximale und die minimale Gesamttotzeit am Trainingsende bilden.

Neuronale Identifikation mit Vorwissen			
Versuchsreihe	**Totzeit**	**Ergebnisintervalle**	**Prozentualer Fehler**
A	$5,\bar{5} \cdot 10^{-4}$	$[5,551\,;5,606]_{max}$	**0,04% – 0,91%**
		$[5,502\,;5,549]_{ave}$	0,11% – 0,95%
		$[5,483\,;5,516]_{min}$	0,71% – 1,29%
B	$2,\bar{7} \cdot 10^{-4}$	$[2,778\,;2,805]_{max}$	**0,01% – 0,98%**
		$[2,745\,;2,773]_{ave}$	0,17% – 1,18%
		$[2,740\,;2,758]_{min}$	0,69% – 1,36%
C	$1,851 \cdot 10^{-4}$	$[1,849\,;1,869]_{max}$	**0,02% – 0,93%**
		$[1,831\,;1,844]_{ave}$	0,39% – 1,09%
		$[1,818\,;1,846]_{min}$	0,28% – 1,79%
D	$1,3\bar{8} \cdot 10^{-4}$	$[1,389\,;1,399]_{max}$	**0,02% – 0,74%**
		$[1,376\,;1,387]_{ave}$	0,13% – 0,91%
		$[1,371\,;1,380]_{min}$	0,57% – 1,26%

Tabelle 7.2: Neuronale Identifikation der Dead-Beat-Totzeiten mit Vorwissen. Die Tabelle faßt die Identifikationsendergebnisse für alle Versuchsreihen zusammen. Im Unterschied zu den Ergebnissen in Tabelle 7.1 wurden in diesen Versuchen die Zeitverzögerungsparameter der EAT-Experten gezielt vorinitialisiert: die Zeitverzögerungsparameter wurden in dem in (7.6) angegebenen Wertebereich zufällig initialisiert. Die Identifikationsendergebnisse bestehen aus den Identifikationsintervallen für die maximale, die durchschnittliche und die minimale Gesamttotzeit. Deutlich zu erkennen ist eine signifikante Verbesserung der Totzeitidentifikation bei allen Versuchsreihen verglichen zu den Ergebnissen ohne Verwendung des Vorwissens. Die prozentualen Fehler zu den Identifikationsintervallen der maximalen Gesamttotzeit sind fett markiert. Zugunsten der Übersichtlichkeit wurde bei den Identifikationsintervallen der Faktor 10^{-4} weggelassen. Abbildung 7.7 auf S. 89 visualisiert ausgewählte Ergebnisse in Identifikationsplots.

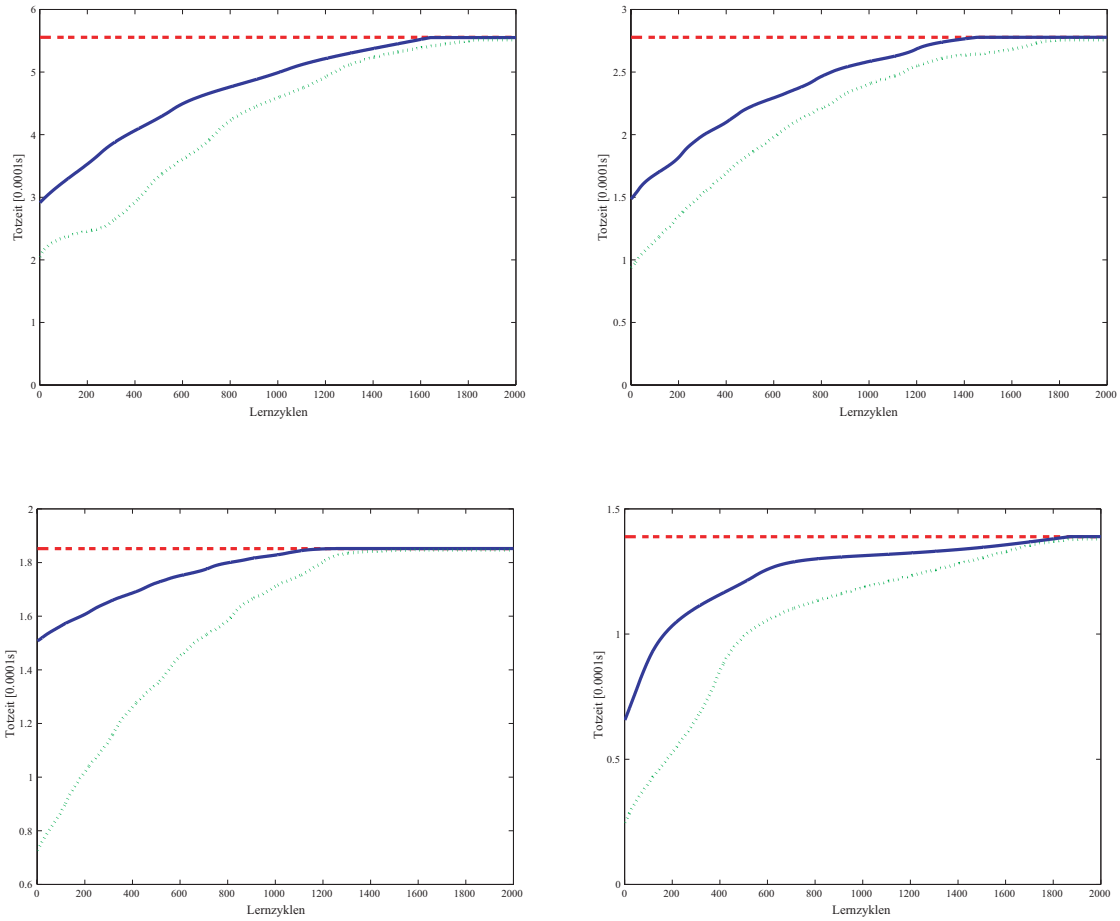

Abbildung 7.7: Identifikationsplots beim nichtprädiktiven Dead-Beat-Verfahren unter Verwendung von Vorwissen. Zu sehen sind Identifikationsplots zu den einzelnen Versuchsreihen. Oben links: Versuchsreihe A. Oben rechts: Versuchsreihe B. Unten links: Versuchsreihe C. Unten rechts: Versuchsreihe D. Die horizontale, rot-gestrichelte Linie gibt jeweils den Wert der zu identifizierenden Totzeit an. Die Identifikationsplots visualisieren die Entwicklung der maximalen (blau-durchgezogen) und der minimalen (grün-gepunktet) Gesamttotzeit der jeweils besten EAT-Experten (aus Tabelle 7.2). Das Einbringen von Vorwissen verbessert die Ergebnisse in mehrfacher Hinsicht. Zum einen konvergieren die Totzeiten nach deutlich weniger Lernzyklen, zum anderen fällt die Breite des Intervalls, das die maximale und die minimale Gesamttotzeit zu Trainingsende bilden, deutlich geringer aus.

Neuronale Identifikation mit verrauschten Daten			
Versuchsreihe	**Totzeit**	**Ergebnisintervalle**	**Prozentualer Fehler**
A	$5,\bar{5} \cdot 10^{-4}$	$[5,392\,;5,899]_{max}$	**2,87% – 6,19%**
		$[5,326\,;5,671]_{ave}$	1,93% – 4,12%
		$[5,130\,;5,317]_{min}$	4,28% – 7,66%
B	$2,\bar{7} \cdot 10^{-4}$	$[2,729\,;2,899]_{max}$	**1,16% – 4,37%**
		$[2,631\,;2,840]_{ave}$	2,25% – 5,26%
		$[2,550\,;2,675]_{min}$	3,67% – 8,19%
C	$1,851 \cdot 10^{-4}$	$[1,894\,;1,928]_{max}$	**2,29% – 4,13%**
		$[1,772\,;1,815]_{ave}$	1,96% – 4,31%
		$[1,707\,;1,794]_{min}$	3,11% – 7,78%
D	$1,3\bar{8} \cdot 10^{-4}$	$[1,403\,;1,441]_{max}$	**1,07% – 3,78%**
		$[1,322\,;1,370]_{ave}$	1,35% – 4,79%
		$[1,261\,;1,349]_{min}$	2,82% – 9,28%

Tabelle 7.3: Neuronale Identifikation der Dead-Beat-Totzeiten mit verrauschten Daten. Die Tabelle faßt die Identifikationsendergebnisse für alle Versuchsreihen zusammen. Im Unterschied zu den Ergebnissen in Tabelle 7.1 wurden bei diesen Versuchen verrauschte Trainingsdaten verwendet. Die Identifikationsendergebnisse bestehen aus den Identifikationsintervallen für die maximale, die durchschnittliche und die minimale Gesamttotzeit. Das Rauschen in den Daten führt zu schlechteren Identifikationsergebnissen. Zum einen sind die prozentualen Fehler im Vergleich zu den Ergebnissen ohne Rauschen erhöht. Zum anderen fallen die Intervalle breiter aus. Dennoch ist die Totzeitidentifikation akzeptabel, so daß die neuronale Identifikationsmethode als robust gegenüber verrauschten Daten betrachtet werden kann. Die prozentualen Fehler zu den Identifikationsintervallen der maximalen Gesamttotzeit sind fett markiert. Zugunsten der Übersichtlichkeit wurde bei den Identifikationsintervallen der Faktor 10^{-4} weggelassen. Abbildung 7.8 auf S. 91 visualisiert ausgewählte Ergebnisse in Identifikationsplots.

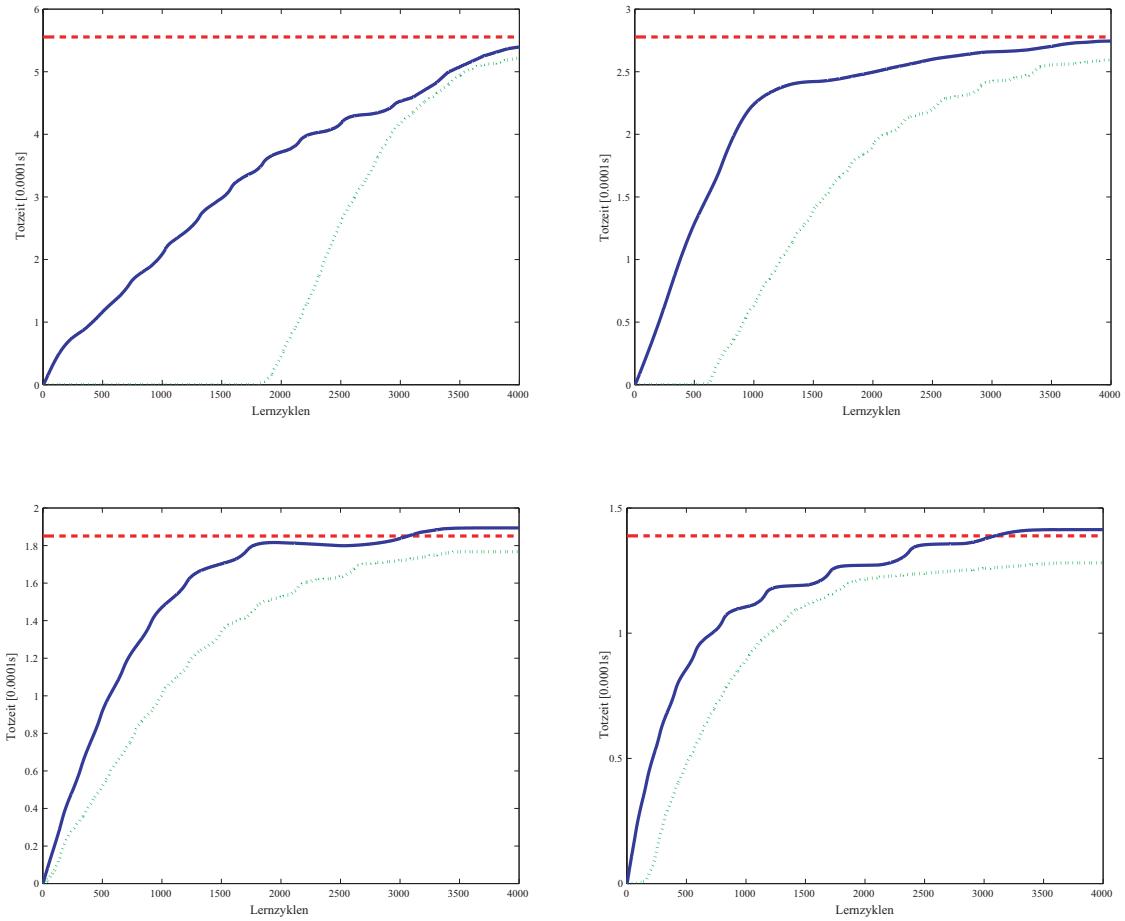

Abbildung 7.8: Identifikationsplots beim nichtprädiktiven Dead-Beat-Verfahren mit verrauschten Daten. Zu sehen sind Identifikationsplots zu den einzelnen Versuchsreihen. Oben links: Versuchsreihe A. Oben rechts: Versuchsreihe B. Unten links: Versuchsreihe C. Unten rechts: Versuchsreihe D. Die horizontale, rot-gestrichelte Linie gibt jeweils den Wert der zu identifizierenden Totzeit an. Die Identifikationsplots visualisieren die Entwicklung der maximalen (blau-durchgezogen) und der minimalen (grün-gepunktet) Gesamttotzeit der jeweils besten EAT-Experten (aus Tabelle 7.3). Die Verwendung verrauschter Daten führt zwar zu schlechteren Identifikationsergebnsissen, aber die Identifikation ist dennoch akzeptabel. Die Netze müssen länger trainiert werden und Distanz zum Zielwert ist größer im Vergleich zu den Versuchen ohne Rauschen.

Neuronale Identifikation bei unterschiedlichen Abtastraten			
Versuchsreihe	**Totzeit**	**Ergebnisintervalle**	**Prozentualer Fehler**
C1	$1,851 \cdot 10^{-4}$	$[1,830\,;1,891]_{max}$	**0,91% – 2,15%**
		$[1,792\,;1,869]_{ave}$	0,87% – 3,21%
		$[1,783\,;1,826]_{min}$	1,37% – 3,69%
C2	$1,851 \cdot 10^{-4}$	$[1,927\,;1,968]_{max}$	**4,07% – 6,28%**
		$[1,710\,;1,778]_{ave}$	3,97% – 7,64%
		$[1,690\,;1,775]_{min}$	4,11% – 8,73%
C3	$1,851 \cdot 10^{-4}$	$[1,959\,;2,039]_{max}$	**5,83% – 10,13%**
		$[1,615\,;1,737]_{ave}$	6,18% – 12,78%
		$[1,568\,;1,706]_{min}$	7,85% – 15,32%

Tabelle 7.4: Neuronale Identifikation der Dead-Beat-Totzeiten bei unterschiedlichen Abtastraten. Die Tabelle faßt die Identifikationsendergebnisse für die Versuchsreihe C zusammen. Die Trainingsdaten wurden mit drei verschiedenen Raten abgetastet. Die Identifikationsendergebnisse bestehen aus den Identifikationsintervallen für die maximale, die durchschnittliche und die minimale Gesamttotzeit. Man erkennt sehr deutlich wie mit zunehmend gröberer Abtastrate die Güte der Identifikation abnimmt. Die prozentualen Fehler zu den Identifikationsintervallen der maximalen Gesamttotzeit sind fett markiert. Zugunsten der Übersichtlichkeit wurde bei den Identifikationsintervallen der Faktor 10^{-4} weggelassen. Abbildung 7.9 auf S. 93 visualisiert ausgewählte Ergebnisse in Identifikationsplots.

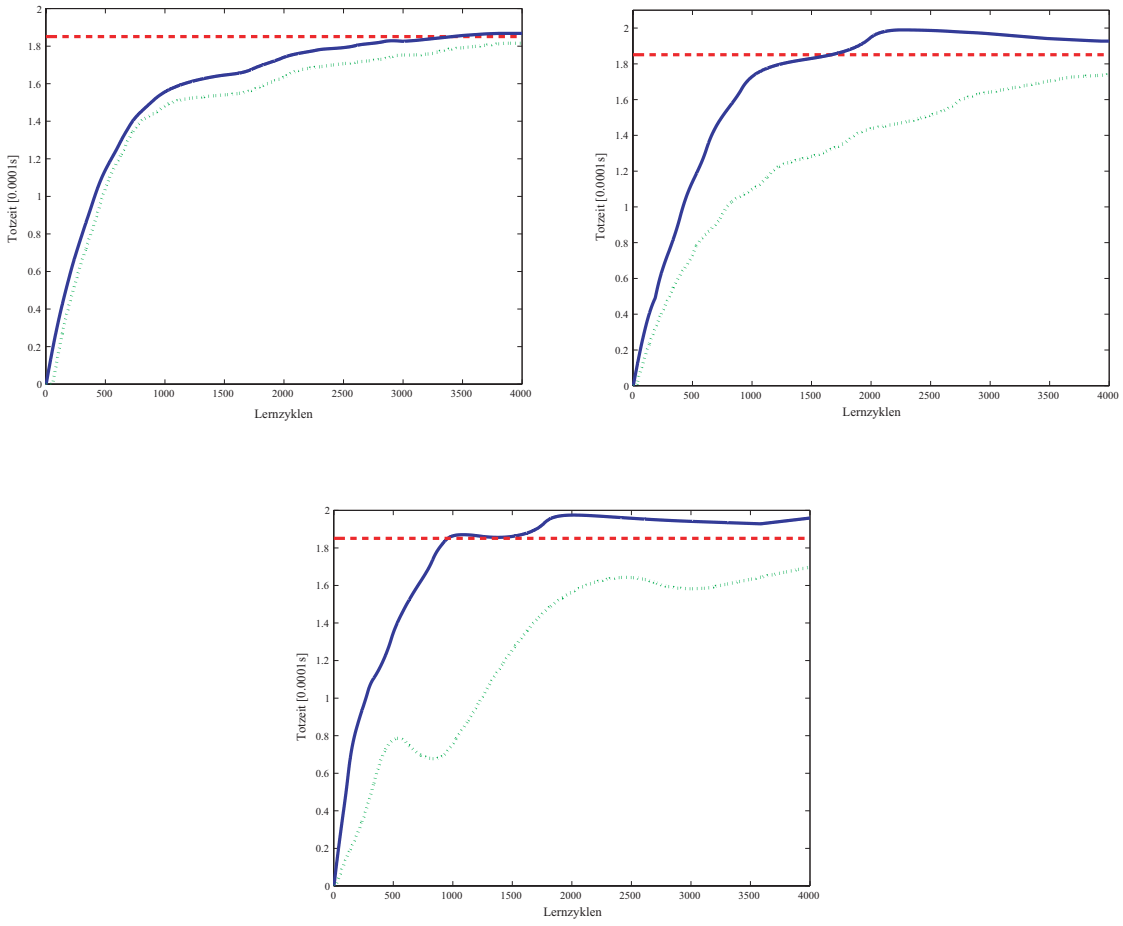

Abbildung 7.9: Identifikationsplots beim nichtprädiktiven Dead-Beat-Verfahren bei Verwendung unterschiedlicher Abtastraten. Zu sehen sind Identifikationsplots zu den einzelnen Versuchsreihen. Oben links: Versuchsreihe C1. Oben rechts: Versuchsreihe C2. Unten: Versuchsreihe C3. Die horizontale, rot-gestrichelte Linie gibt jeweils den Wert der zu identifizierenden Totzeit an. Die Identifikationsplots visualisieren die Entwicklung der maximalen (blau-durchgezogen) und der minimalen (grün-gepunktet) Gesamttotzeit der jeweils besten EAT-Experten (aus Tabelle 7.4). Man erkennt sehr deutlich wie mit zunehmend gröberer Abtastrate die Güte der Identifikation abnimmt.

Neuronale Identifikation zeitvarianter Totzeiten			
Versuchsreihe	**Totzeit**	**Ergebnisintervalle**	**Prozentualer Fehler**
AB	$5,\bar{5} \cdot 10^{-4}$ $2,\bar{7} \cdot 10^{-4}$	$[5,318\,;5,441]_{max}$ $[2,823\,;2,892]_{min}$	2,06% – 4,27% 1,66% – 4,14%
CD	$1,851 \cdot 10^{-4}$ $1,3\bar{8} \cdot 10^{-4}$	$[1,792\,;1,823]_{max}$ $[1,409\,;1,443]_{min}$	1,53% – 3,18% 1,48% – 3,91%

Tabelle 7.5: Neuronale Identifikation zeitvarianter Dead-Beat-Totzeiten. Die Tabelle faßt die Identifikationsendergebnisse für die Versuchsreihen AB und CB zusammen. Angegeben sind jeweils die Identifikationsintervalle für die maximale und die minimale Gesamttotzeit. Zugunsten der Übersichtlichkeit wurde bei den Identifikationsintervallen der Faktor 10^{-4} weggelassen. Abbildung 7.10 visualisiert ausgewählte Ergebnisse in Identifikationsplots.

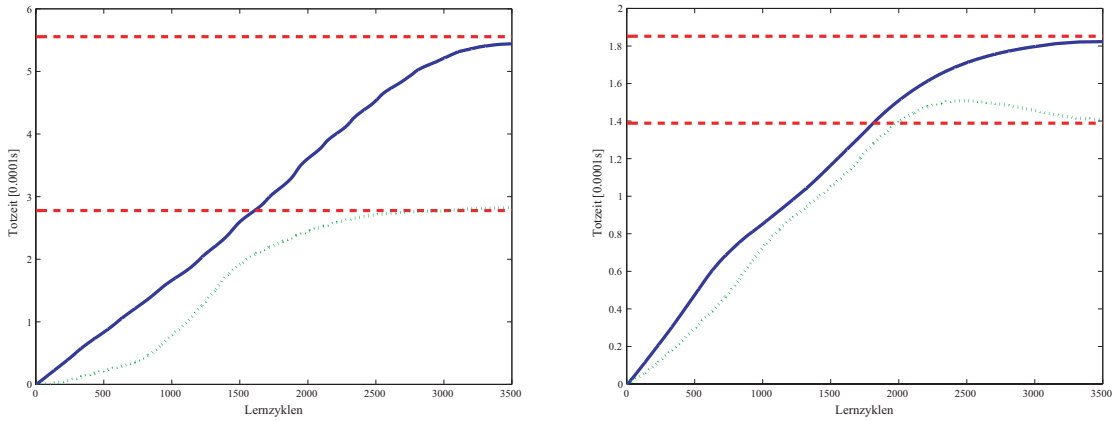

Abbildung 7.10: Identifikationsplots bei zeitvarianten Totzeiten. Zu sehen sind Identifikationsplots zu den einzelnen Versuchsreihen. Links: Versuchsreihe AB. Rechts: Versuchsreihe CD. Die horizontalen, rot-gestrichelten Linien geben jeweils die Grenzen des Wertebereichs der zeitvarianten Totzeit an. Die Identifikationsplots visualisieren die Entwicklung der maximalen (blau-durchgezogen) und der minimalen (grün-gepunktet) Gesamttotzeit der jeweils besten EAT-Experten (aus Tabelle 7.5). Man erkennt wie die maximale Gesamttotzeit gegen die obere und die minimale Gesamttotzeit gegen die untere Bereichsgrenze konvergiert.

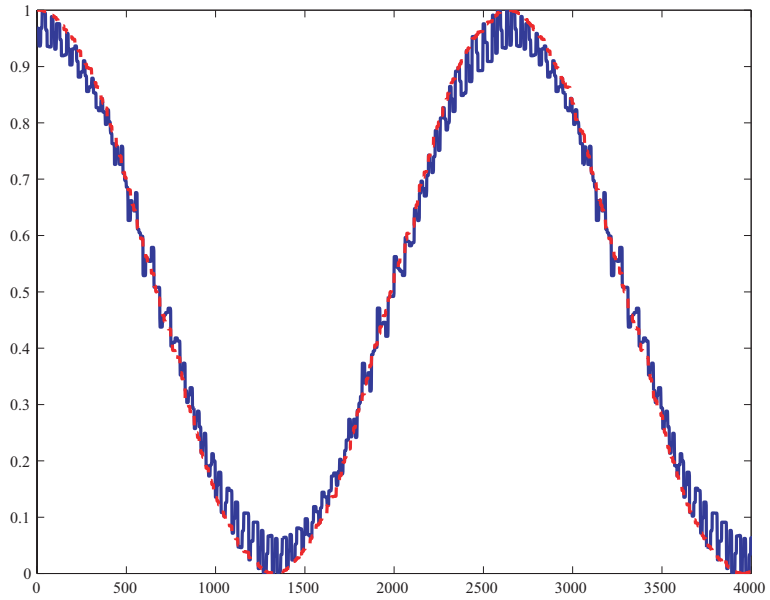

Abbildung 7.11: Trainingsdaten der Versuchsreihe B ($F_n = 50\text{Hz}$) beim prädiktiven Dead-Beat-Verfahren. Zu sehen ist ein Ausschnitt aus dem normierten Trainingsdatensatz. Die rot-gestrichelte Linie zeigt den Verlauf des Sollstroms, die blau-durchgezogene Linie den Verlauf des Iststroms.

Neuronale Identifikation im prädiktiven Fall		
Versuchsreihe	**Totzeit**	**Ergebnisintervalle**
A	*keine*	$[0,00218 \cdot 10^{-4}\,;0,00341 \cdot 10^{-4}]_{max}$ $[0,00196 \cdot 10^{-4}\,;0,00287 \cdot 10^{-4}]_{ave}$ $[0\,;0,00062 \cdot 10^{-4}]_{min}$

Tabelle 7.6: Neuronale Identifikation der Dead-Beat-Totzeiten im prädiktiven Fall. Die Tabelle faßt exemplarisch die Identifikationsendergebnisse für die Versuchsreihe A zusammen. Die Identifikationsendergebnisse bestehen aus den Identifikationsintervallen für die maximale, die durchschnittliche und die minimale Gesamttotzeit. Man erkennt deutlich, daß keine Totzeit identifiziert wurde.

Kapitel 8

Experimente zur Regelung von Totzeitsystemen

Das vorherige Kapitel zeigte die erfolgreiche Identifikation von Totzeiten in nichtlinearen dynamischen Systemen. In diesem Kapitel werden zwei Anwendungsbeispiele vorgestellt, bei denen unter Verwendung des in Kapitel 6 vorgestellten neuronalen Konzepts eine verbesserte Regelung erreicht wird.

Das erste Beispiel kommt aus der Automobilindustrie und behandelt einen Prüfstand für Ottomotoren. Das zweite Beispiel stammt aus der Lebensmittelindustrie und zeigt die Regelung der Dampfzufuhr bei einer Mischanlage. Beide Anwendungen liegen als Simulationen in Matlab/Simulink vor. Matlab/Simulink ist eine wissenschaftlich-technische Software für die Simulation dynamischer Systeme aus den Gebieten der technischen Mechanik, Elektrotechnik und Regelungstechnik. Neben den Handbüchern [74] und [73] ist als anschauliche Einführung in dieses Werkzeug das Lehrbuch [28] zu empfehlen. Simulationen technischer Vorgänge spielen eine zunehmend wichtige Rolle, da sie Lösungen validieren und den Prozeß des Auffindes neuer Lösungen unterstützen. Die Modellbibliothek AMoC[1] stellt daher eine Anbindung an Matlab/Simulink bereit, die es ermöglicht, die in der Bibliothek implementierten Modelle in Matlab/Simulink-Simulationen zu verwenden [72].

Die Anregung zur Auseinandersetzung mit dem Anwendungsbeispiel Prüfstand stammt von einem Projektpartner im BMBF-Verbundprojekt ACON, der Kratzer Automation AG. In Zusammenarbeit mit Kratzer Automation entstand die in Abschnitt 8.1.1 vorgestellte Simulation eines realen Motorenprüfstands. Der Firma Kratzer Automation und ihren Mitarbeitern — namentlich Dirk Stübener und Dr. Klaus Eder — sei hiermit für die Zusammenarbeit gedankt.

[1] ACON Model Classes: Implementierung diverser neuronaler Modelle in einheitlicher Sichtweise in C++, siehe auch S. 31

Die Anregung für das zweite Anwendungsbeispiel, der Mischanlage aus der Lebensmittelindustrie, stammt aus einer wissenschaftlichen Zusammenarbeit mit dem Lehrstuhl für Regelungstechnik am Institut für Elektrotechnik der Technischen Universität Prag. Für die Bereitstellung der Simulation der Mischanlage in Matlab/Simulink sei Dr. Petr Horáček an dieser Stelle gedankt.

Im Folgenden wird für jedes der beiden Anwendungsbeispiele zunächst eine Systembeschreibung gegeben, daran anschließend wird die verwendete Regelungsstrategie erläutert und schließlich die verbesserten Regelungsergebnisse gezeigt.

8.1 Motoren-Prüfstand

In diesem Abschnitt wird dargelegt, wie unter Verwendung des EAT-Modells eine verbesserte Regelung in einer Motorenprüfstandsumgebung erzielt wird[2].

8.1.1 Systembeschreibung

Die Entwicklung von Motoren oder Autobauteilen, wie beispielsweise Getriebe oder Katalysator, geht einher mit Tests der entwickelten Anteile am Prüfstand. Bei der Regelung von Verbrennungsmotoren kommt es dabei darauf an, ein gegebenes Fahrprofil mit hoher Genauigkeit nachzufahren. Für die Versuche mit neuronalen Reglerentwürfen wurde eine Prüfstand-Simulation in Matlab/Simulink erstellt, deren schematische Darstellung in Abbildung 8.1 zu sehen ist.

Das Modell bildet einen realen Motorenprüfstand nach. Es besteht aus einem Ottomotor als Prüfling, einer E-Maschine zur Lasterzeugung, einem Fahrzeugmodell zur Lastvorgabe und einem Regler zur Geschwindigkeitsregelung. Der Ottomotor wird durch ein Mittelwertsmodell beschrieben. Das Fahrzeugmodell simuliert das reale Auto mit Roll- und Fahrwiderstand und Bremsen. Die E-Maschine erhält vom Fahrzeugmodell über das Drehmoment eine Vorgabe zur Lasterzeugung. Damit wird der Einfluß des realen Fahrzeugs auf den Motor simuliert. Der grau hinterlegte Bereich entspricht den Hardware-Komponenten eines realen Prüfstands. Eine ausführliche Beschreibung der Simulation findet sich in [78].

Der Regler wird versorgt mit der Referenzgeschwindigkeit des Fahrprofils, welches als Tabelle in der Fahrzeugsimulation vorliegt, und mit der aktuellen Geschwindigkeit. Aus diesen Größen wird der Regelfehler berechnet. Der Regler gibt dann die Gaspedalstellung $\alpha \in [0, 90^\circ]$ vor, welche vom Pedalwertgeber eingestellt wird.

[2]Die hier vorgestellten Arbeiten führten zu den Vorabveröffentlichungen [78] und [77].

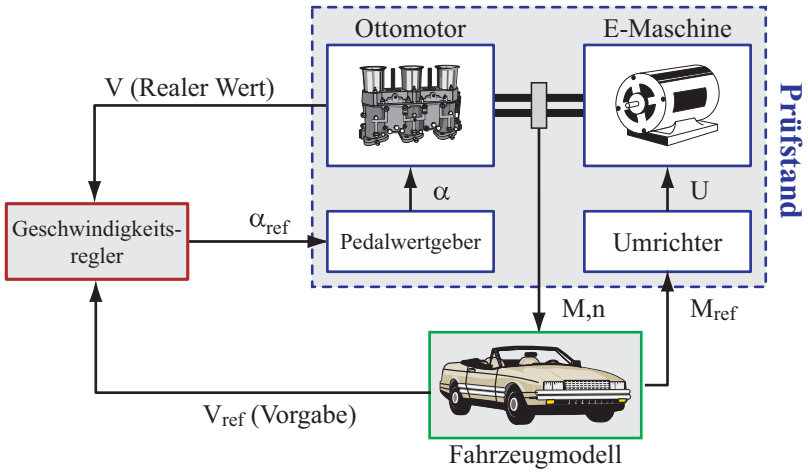

Abbildung 8.1: Simulation in Matlab/Simulink eines Prüfstand bestehend aus einem Ottomotor als Prüfling, einer E-Maschine zur Lasterzeugung und einem Fahrzeugmodell zur Lastvorgabe. Ein Regler gibt die Gaspedalstellung vor und regelt damit die Fahrgeschwindigkeit.

Derzeit basiert die Motorregelung auf der Verwendung von inversen Kennfeldern und herkömmlichen PI-Reglern. Die inversen Kennfelder werden dazu benutzt, die Gaspedalstellung abhängig vom gewünschten Drehmoment vorherzusagen. Das Problem bei diesem Ansatz ist der hohe Meßaufwand, da der gesamte Prozeßraum hinreichend genug abgefahren werden muß, was den Motor stark belastet.

8.1.2 PINN-Regelung

Als Regelungsstrategie wird eine Regelung mit neuronaler Vorsteuerung angewandt (vgl. Abschnitt 4.4.2, S. 36). Da der Ottomotor zu den anspruchsvollen Regelstrecken zählt, bei denen Totzeiten zu berücksichtigen sind (vgl. Abschnitt 3.2), wird das in Kapitel 6 vorgestellte EAT-Netz als neuronale Vorsteuerung verwendet. Damit wird die in Abschnitt 6.1 erwähnte Möglichkeit verfolgt, ein trainiertes EAT-Netz als Prozeßmodell innerhalb eines intelligenten Reglers einzusetzen. Eine explizite Interpretation des Netzes ist bei diesem Ansatz nicht nötig.

Abbildung 8.2 veranschaulicht den verwendeten neuro-gestützten Regler. Es handelt sich um einen erweiterten PI-Regler. Das EAT-Netz dient als inverses Prozeßmodell und ist in der Abbildung mit NN bezeichnet. Diese neuronale Komponente wird in den Regler als dritte Komponente neben dem Proportional- und Integralanteil eingefügt. Der Begrenzungs- und Anti-Windup-Mechanismus ist dabei hinter dem neuronalen Aufschaltungspunkt plaziert, so daß die Netzvorhersagen von diesen Mechanismen mit berücksichtigt werden.

Aufgrund seiner Topologie wird dieser Regler als *PINN-Regler* bezeichnet [78]. Mit einem guten inversen Prozeßmodell wird der PI-Regler zu Null gefahren, da das Netz seine Funktion übernimmt. Dennoch verbleibt der PI-Regler als Sicherungs-Mechanismus und regelt im Falle unvorhergesehener Netzaussagen den verursachten Fehler aus. Die statischen inversen Kennfelder werden also durch dynamische neuronale Netze ersetzt.

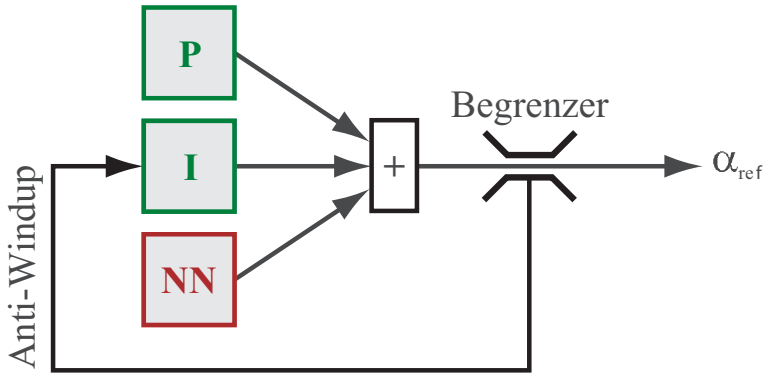

Abbildung 8.2: Veranschaulichung der PINN-Regelung. Ein herkömmlicher PI-Regler wird durch die neuronale Komponente NN, einem inversen Prozeßmodell, unterstützt.

8.1.3 Ergebnisse und Bewertung

Die Regelungsaufgabe besteht nun darin, einem gegebenen Fahrprofil möglichst exakt zu folgen. Mit dem vorgestellten PINN-Regler soll dabei ein besseres Ergebnis erzielt werden als mit dem herkömmlichen Kennfeld-Ansatz. Das Ergebnis wird gemessen als der quadratische Fehlerabstand der gesamten gefahrenen Geschwindigkeitstrajektorie $\vec{v}$ von der Sollvorgabe, dem gegebenen Fahrprofil $\vec{v}_{ref}$. Dabei gilt eine Performanzverbesserung um mindestens 5% als erwähnenswert.

Es wurden zwei unterschiedliche Simulationen verwendet. Simulation A bezeichnet die in Abschnitt 8.1.1 beschriebene Simulation mit einem Ottomotor-Mittelwertmodell als Prüfling. In Simulation B wurde zusätzlich der Momentenverlauf mittels einer Oberschwingung verrauscht, so daß Simulation B im Vergleich zu Simulation A eine realistischere Simulation mit Meßrauschen darstellt. Abbildung 8.3 zeigt das in beiden Simulationen verwendete Fahrprofil.

Wie bereits erwähnt wird das EAT-Netz als inverses Prozeßmodell verwendet. Die Trainingsdaten wurden erzeugt, indem die Simulationen mit dem vordefinierten Fahrprofil betrieben wurden und dabei die Eingabe- und Ausgabegrößen für das EAT-Netz erfaßt wurden. Als Eingaben dienen dem Netz die Systemgrößen Drehmoment und Motorge-

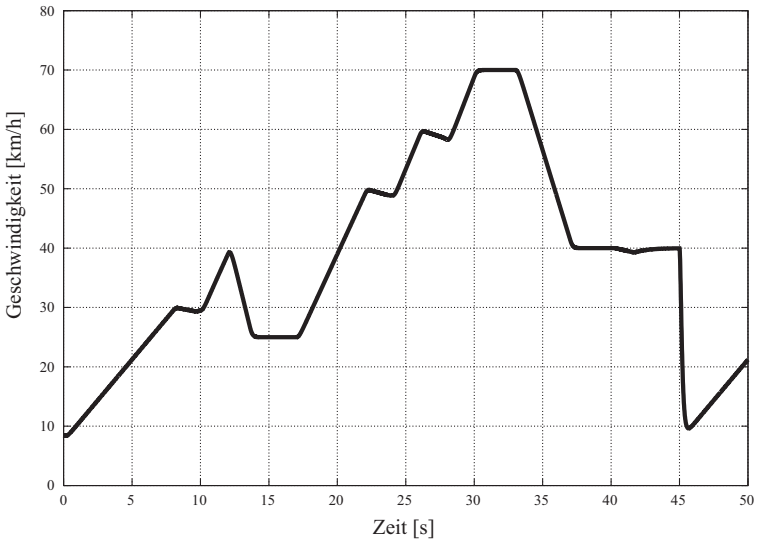

Abbildung 8.3: Das Fahrprofil der Prüfstandsimulation

schwindigkeit und als Zielwert das gemessene Stellsignal des Reglers. Die Messungen wurden über einen Zeitraum von 50 Sekunden erfaßt, ein Datenpunkt alle 0.01 Sekunden. Die Eingaben sowie der Zielwert wurden auf das Intervall $[0; 1]$ normiert. Aus 100 Zufallsinitialisierungen wurde das beste EAT-Netz ausgewählt und mit den Trainingsdaten trainiert. Die beste Initialisierung bezeichnet dabei diejenige mit dem kleinsten Netzfehler nach einmaliger Präsentation der Trainingsdaten. Durch diese einfache Zufallssuche wird eine günstige Startposition für die im Training durchgeführte Optimierung gewonnen. Das trainierte Netz wird dann innerhalb der Simulation als NN-Komponente des PINN-Reglers verwendet.

Tabelle 8.1 zeigt die prozentuale Verbesserung der Performanz bei Verwendung der PINN-Reglung mit verschiedenen dynamischen neuronalen Modellen als NN-Komponente im Vergleich zur herkömmlichen PI-Regelung. Wie man erkennt, führt die Verwendung des EAT-Netzes zum besten Ergebnis. Bei Simulation A übersteigt es als einziges Modell die kritische 5%-Performanzmarke. Bei der verrauschten Simulation B erreichen alle Modelle bessere Ergebnisse, wobei auch hier das EAT-Netz an der Spitze steht. Offensichtlich konnte das EAT-Netz aufgrund seiner adaptierbaren Zeitverzögerungsparameter in Kombination mit einem flexiblen Lernverfahren die Dynamik dieses Totzeitsystems am besten nachbilden.

Das EAT-Netz und die beiden anderen Modelle ATNN und DRBF wurden als globale Modelle eingesetzt, d.h. es wird ein Modell für den gesamten (inversen) Prozeß verwendet. Einen anderen Ansatz bildet die lokale Modellierung. In der Dissertation [71] wird der lokal-ellipsoide Modellierungsansatz LEMON vorgestellt. Der Prozeßraum wird in ellipsoi-

Modell	Verbesserung	
	Simulation A	Simulation B
ATNN	2.9%	4.5%
DRBF	1.3%	7.1%
EAT-Netz	6.1%	8.6%

Tabelle 8.1: Prozentuale Verbesserung der Performanz durch Einsatz des PINN-Reglers unter Verwendung des EAT-Netzes im Vergleich zu anderen dynamischen neuronalen Modellen. Simulation B unterscheidet sich von Simulation A durch ein zusätzliches Meßrauschen im Momentenverlauf. Bei beiden Simulationen zeigt das EAT-Netz die beste Performanz.

de Teilbereiche zerlegt und jedem Ellipsoid ein lokales (lineares oder statisch neuronales) Modell zugeordnet, das den Prozeß in diesem Zuständigkeitsbereich modelliert. Dieser Ansatz ermöglicht somit die Modellierung komplexer Systeme durch das Zusammenwirken von mehreren lokalen Modellen, die auf bestimmte Teilbereiche spezialisiert sind. In [71] wurde das gleiche Anwendungsbeispiel verwendet und LEMON als NN-Komponente des PINN-Reglers eingesetzt. Dabei erreichte dieser Ansatz prozentuale Verbesserungen von 9.0% bzw. 10.9% bei den Simulationen A bzw. B. Dies ist im Vergleich zum EAT-Netz zwar eine bessere Performanz, allerdings sind eine wesentlich höhere Modellkomplexität und damit ein erhöhter Trainings- und Simulationsaufwand zu berücksichtigen.

8.2 Lebensmittel-Mischanlage

Dieser Abschnitt stellt eine Anwendung aus der Lebensmittelindustrie vor. Es wird gezeigt, wie die in Kapitel 6 vorgestellte neuronale Identifikation von Totzeiten als Basis für die Regelung der Dampfzufuhr bei einer Mischanlage genutzt werden kann.

8.2.1 Systembeschreibung

Bei der Herstellung von Lebensmitteln werden Mischanlagen unterschiedlicher Art verwendet, um von den einzelnen Zutaten zum Endprodukt zu gelangen. Im vorliegenden Beispiel geht es um die Vermengung von Trockenzutaten mit Wasser unter Erhitzung des Gemisches durch Dampfzufuhr. Abbildung 8.4 zeigt schematisch den Aufbau der betrachteten Mischanlage.

Die Trockenzutaten gelangen über ein Förderband, das Wasser über ein Rohr in den Mischbehälter. Das Trockenzutaten-Wasser-Gemisch wird im Mischbehälter mittels zweier

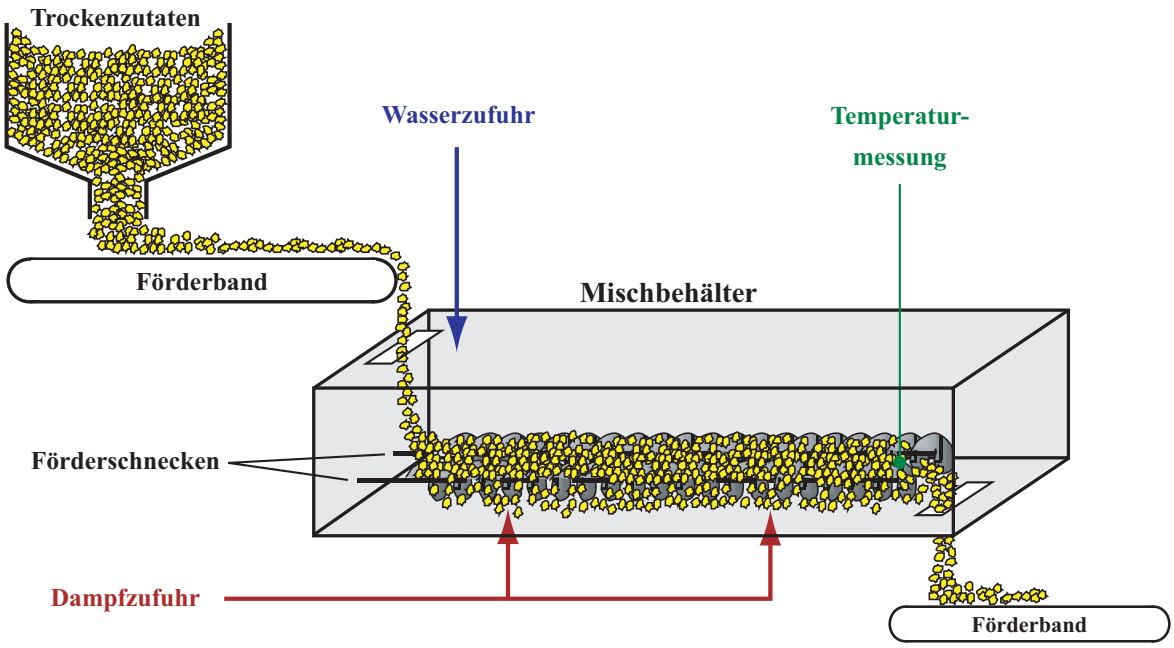

Abbildung 8.4: Darstellung einer Lebensmittel-Mischanlage zur Vermengung von Trockenzutaten mit Wasser unter Dampferhitzung. Geregelt werden soll die Dampfzufuhr, so daß im Mischbehälter die Temperatur konstant gehalten wird.

Förderschnecken mit konstanter Geschwindigkeit befördert und gleichzeitig vermengt. An zwei Stellen am Boden des Mischbehälters wird der Mischanlage Wasserdampf zugeführt und dadurch das Gemisch erhitzt. Die beiden Dampfeinlässe werden über ein gemeinsames Ventil gesteuert. Am Mischbehälterausgang gelangt das Gemisch auf ein Förderband, auf dem es zur Weiterverarbeitung transportiert wird. Das Endprodukt dieser Mischanlage ist eine feucht-warme Masse, die sich in Geschmack, Geruch und evtl. auch in der Farbe von dem Trockenzutatengemisch unterscheidet. Die neue Konsistenz des Gemisches ermöglicht die Weiterverarbeitung in Pressen, um zur gewünschten Form des Endprodukts zu gelangen. Diese Mischanlage ist beispielsweise als Teil des Herstellungsprozesses von Müsliriegeln denkbar. Eine genaue Beschreibung der Simulation in Matlab/Simulink, die eine derartige reale Mischanlage nachbildet, findet sich in [30].

Die Regelungsaufgabe in diesem Anwendungsbeispiel ist die Regelung der Dampfzufuhr, so daß die Temperatur im Mischbehälter konstant gehalten wird. Die Sollwertvorgabe für die Temperatur hängt von der verwendeten Rezeptur der Trockenzutaten ab und liegt zwischen 80-150 Grad Fahrenheit (dies entspricht 26,6 – 65,5 Grad Celsius). Die aktuelle Temperatur im Mischbehälter wird durch einen Sensor kurz vor dem Ausschüttungsschlitz gemessen.

Bei diesem Mischanlagenprozeß handelt es sich um einen nichtlinearen Prozeß mit signi-

fikanter Totzeit. Die Totzeit wird verursacht durch die verzögerte Messung der Temperatur und entsteht auf der Strecke vom Dampfventil zum Temperatursensor. Während die Dampfzufuhr die Temperatur erhöht, sorgt der Einlauf der Trockenzutaten und des Wassers, sowie der Auslauf der Fertigmasse für Kühlung. Auch Abstrahlungseffekte, d.h. die Umgebungstemperatur sind zu berücksichtigen. Die Totzeit ist in ihrer Größenordnung bekannt und bewegt sich zwischen ca. 25-35 Sekunden je nach applizierter Rezeptur des Trockengemisches.

8.2.2 Regelungsstrategie

Die Regelung erfolgt bislang manuell durch einen menschlichen Experten. Die Umstellung auf eine automatisierte Regelung erfordert die Festlegung einer geeigneten Regelstrategie. Eine herkömmliche PI oder PID-Regelung versagt in dieser Anwendung aufgrund der signifikanten Totzeit, die von diesen Reglern nicht berücksichtigt werden kann. Das Ergebnis einer PI-Regelung zeigt Abbildung 8.5. Zu sehen ist der Verlauf der Mischanlagentemperatur bei einer Sollwertvorgabe von 150 Grad Fahrenheit. Die Instabilität dieser Art von Regelung ist in den deutlichen Temperaturschwankungen erkennbar.

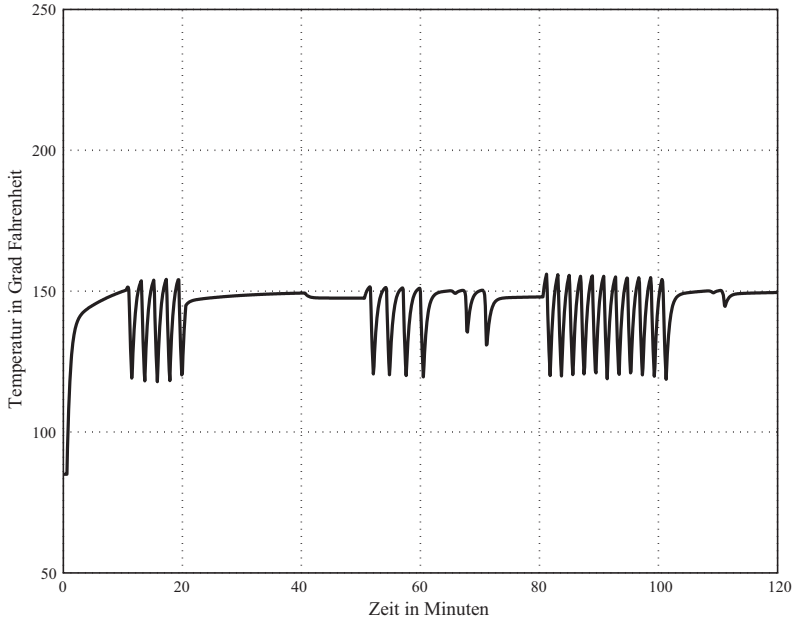

Abbildung 8.5: Verlauf der Temperatur im Mischbehälter bei Verwendung einer herkömmlichen PI-Regelung. Zu erkennen sind deutliche Temperaturschwankungen, welche auf eine instabile Regelung zurückzuführen sind.

Eine PINN-Regelung mit einem EAT-Netz als NN-Komponente, wie sie im vorherigen Anwendungsbeispiel Prüfstand in Abschnitt 8.1 erfolgreich eingesetzt wurde, bewirkt in diesem Beispiel keine nennenswerte Verbesserung — es konnte lediglich eine prozentuale Verbesserung von ca. 1% erreicht werden. In [30] wird ein Ansatz vorgestellt, der mit Hilfe einer Fuzzy-Regelbasis die Parameter eines PI-Reglers abhängig vom Systemzustand verändert. Der Aufbau einer adäquaten Fuzzy-Regelbasis allerdings erscheint in diesem Zusammenhang schwierig. Daher wird im Folgenden eine alternative Strategie verfolgt.

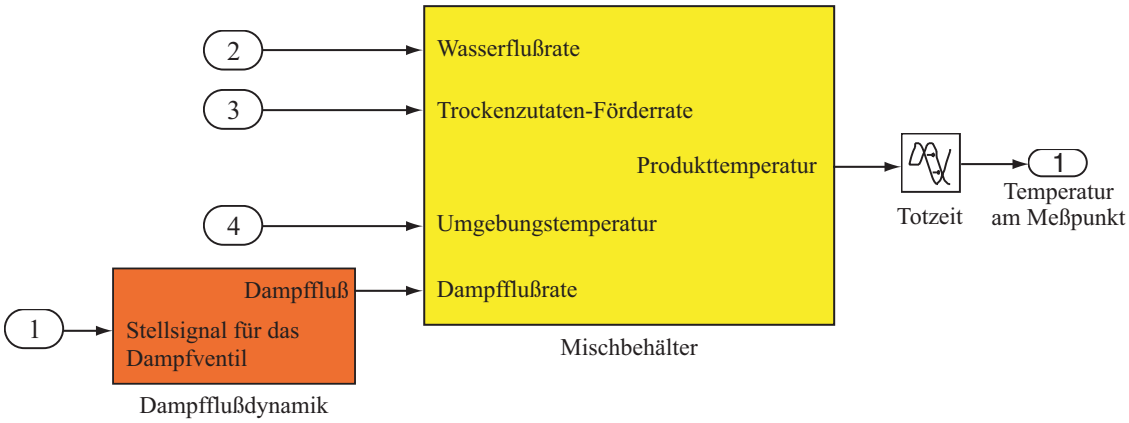

Abbildung 8.6: Ausschnitt aus der Matlab/Simulink-Simulation der Mischanlage.

Abbildung 8.6 zeigt einen Ausschnitt aus der Matlab/Simulink-Simulation der Mischanlage. Zu sehen ist die Modellierung des Mischanlagenbehälters als Hintereinanderschaltung des dynamischen Verhaltens des Behälters mit der vorhandenen Totzeit. Dies impliziert die Verwendung des in Abschnitt 3.4 vorgestellten Regelprinzips nach Smith (vgl. S. 25). Dieses Regelprinzip verwendet ein Modell des totzeitfreien Systems kombiniert mit einer Totzeitkompensation. Die im Smith-Regler verwendete Totzeit wird im Folgenden als Modelltotzeit bezeichnet im Unterschied zur tatsächlichen Systemtotzeit. Die Systemtotzeit ist lediglich in ihrer ungefähren Größenordnung bekannt und das verwendete Regelprinzip ist sehr empfindlich gegenüber Unterschieden der Systemtotzeit und der Modelltotzeit. Daher ist zunächst eine Identifikation der Totzeit notwendig. Das Ergebnis der Identifikation wird dann im Smith-Regler als Modelltotzeit verwendet. Als Regler für die nichtlineare, totzeitfreie Strecke wird ein PINN-Regler (vgl. Abschnitt 8.1.2) mit einem Multilayer-Perzeptron als NN-Komponente verwendet. Abbildung 8.7 zeigt den resultierenden Aufbau des eingesetzten PINN-Smith-Reglers.

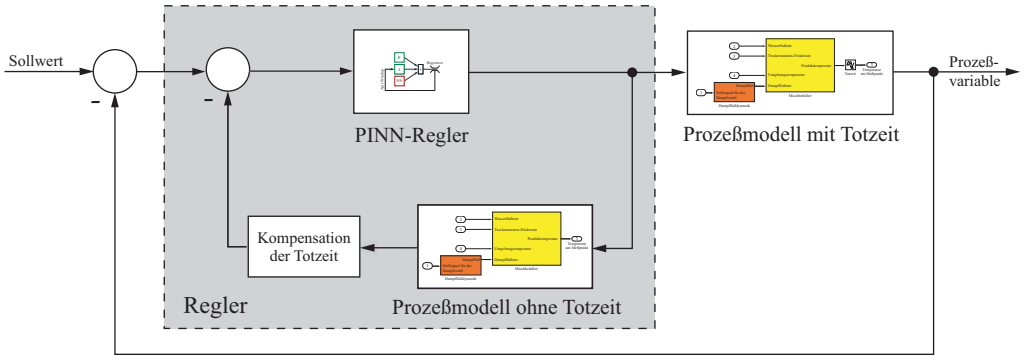

Abbildung 8.7: Regler für die Mischanlage nach dem Prinzip von Smith unter Verwendung eines PINN-Reglers.

8.2.3 Ergebnisse und Bewertung

Die Aufgabenstellung in diesem Anwendungsbeispiel ist die Regelung der Dampfzufuhr, so daß die Temperatur im Mischbehälter der Anlage konstant bleibt. Die Solltemperatur wird vorgegeben und bewegt sich je nach Rezeptur im Intervall $[80; 150°F]$ ($[26,6; 65,5°C]$). Lediglich leichte Schwankungen um die vorgegebene Solltemperatur sind zulässig. Mit der vorgestellten Regelungsstrategie (vgl. Abbildung 8.7) wird das gewünschte Regelziel erreicht. Im Folgenden wird ein Beispielszenario einer Rezeptur mit Solltemperatur $150°F$ vorgestellt.

Zunächst erfolgt die Identifikation der Systemtotzeit. Wie in Abschnitt 6.1 dargelegt (vgl. auch Abbildung 6.1 auf S. 52) bildet die Erfassung eines Trainingsdatensatzes den ersten Schritt. Die Trainingsdaten wurden erzeugt, indem die Matlab/Simulink-Simulation einmal betrieben wurde und dabei die Ein- und Ausgabegrößen für die EAT-Experten erfaßt wurden. Als Eingaben dienen dem Netz die Wasserflußrate, die Trockenzutaten-Föderrate und die Umgebungstemperatur. Die gemessene Temperatur im Mischbehälter bildet den Zielwert. Die Messungen wurden über einen Zeitraum von 120 Minuten erfaßt. Die Eingaben und der Zielwert wurden auf das Intervall [0;1] normiert.

Mit diesen Trainingsdaten wurden 10 EAT-Experten trainiert. Wie in den Versuchen in Kapitel 7 wird auch hier eine einfache Zufallssuche appliziert, um einen guten Startpunkt für die Optimierung im Training zu gewinnen. Aus jeweils 100 Zufallsinitialisierungen wird die jeweils Beste als EAT-Experte ausgewählt. Die beste Initialisierung ist dabei diejenige mit dem kleinsten Netzfehler nach einmaliger Präsentation der Trainingsdaten. Die verwendeten EAT-Netze sind schichtenweise vollvernetzt mit $k = 1$ für jede Verbindung und bestehen jeweils aus drei Eingabeneuronen, zehn Neuronen in einer versteckten Schicht und einem Ausgabeneuron. Die Gewichte wurden zufällig im Intervall [-1;1] initialisiert. Bei den Zeitverzögerungen wird die Tatsache ausgenutzt, daß Vorwissen über die System-

totzeit in Form eines Wertebereichs vorliegt. Die Zeitverzögerungsparameter werden daher wie in Kapitel 6.3.3 eingeführt im Intervall $[\frac{25}{2}; \frac{35}{2}]$ zufällig initialisiert.

Nach dem Training stehen die EAT-Experten als Modelle der Mischanlage zur Verfügung und können nun durch die in Abschnitt 6.3 vorgestellte Methode interpretiert werden, um zur identifizierten Totzeit zu gelangen. Als Ergebnis der Interpretation ergeben sich die in Tabelle 8.2 angegebenen Identifikationsintervalle. Abbildung 8.8 auf S. 109 zeigt die Entwicklung der maximalen Gesamtverzögerung im zugehörigen Identifikationsplot. Anhand dieses Identifikationsergebnisses wird eine Modelltotzeit von 30 Sekunden festgelegt, welche im nächsten Schritt in die im vorherigen Abschnitt beschriebene PINN-Smith-Regelung integriert wird.

Gesamttotzeit	**Identifikationsintervall**
Maximal	$[29,99\,;\,30,03]$
Durchschnittlich	$[29,96\,;\,30,01]$
Minimal	$[29,91\,;\,29,98]$

Tabelle 8.2: Identifikationsintervalle zur Identifikation der Totzeit der Lebensmittelmischanlage. Die Tabelle gibt die Identifikationsintervalle für die minimale, durchschnittliche und maximale Gesamttotzeit an.

Die PINN-Smith-Regelung besteht aus einem PINN-Regler für die totzeitfreie Strecke und einer Totzeitkompensation. Bezeichne τ_M die identifizierte Modelltotzeit, dann lautet die Übertragungsfunktion $G(s)$ des Kompensationsblockes analog zum in Abschnitt 3.4 vorgestellten Smith-Prinzip:

$$G(s) = 1 - e^{-s\tau_M} \tag{8.1}$$

Als neuronale Komponente im PINN-Regler dient ein Multilayer-Perzeptron, das analog zur Prüfstandsanwendung in Abschnitt 8.1 als inverses Modell der Mischanlage trainiert wird. Dazu wurde die Systemausgabe des totzeitfreien Systems, die Temperatur im Mischbehälter, als Netzeingabe und das Stellsignal als Zielwert verwendet. Beide Größen wurden während eines Simulationslaufs unter Regelung eines PI-Reglers erfaßt und daran anschließend auf das Intervall [0;1] normiert. Wie bei vorangehenden Versuchen wurde durch 100 Zufallsinitialisierungen eine einfache Zufallssuche zum Auffinden der besten Initialisierung angewandt.

Im letzten Schritt wird der so entworfene PINN-Smith-Regler zur Regelung der Dampfzufuhr in die Mischanlagensimulation integriert. Abbildung 8.9 auf S. 110 zeigt den nahezu

stabilen Verlauf der Temperatur im Mischbehälter entlang dem gewünschten Sollwert von 150 Grad Fahrenheit. Damit stellt der PINN-Smith-Regler eine erfolgreiche Lösung zur Regelung der Lebensmittelmischanlage dar. Insbesondere wurde eine deutliche Verbesserung gegenüber der Regelung mit einem PI-Regler erreicht (vgl. Abbildung 8.5).

Nachdem in diesem Kapitel die erfolgreiche Anwendung des in dieser Arbeit entwickelten neuronalen Modellierungs- und Identifikationskonzepts anhand von zwei Anwendungen aus unterschiedlichen Industriebereichen gezeigt wurde, folgt im nächsten Kapitel eine Zusammenfassung der Ergebnisse dieser Arbeit.

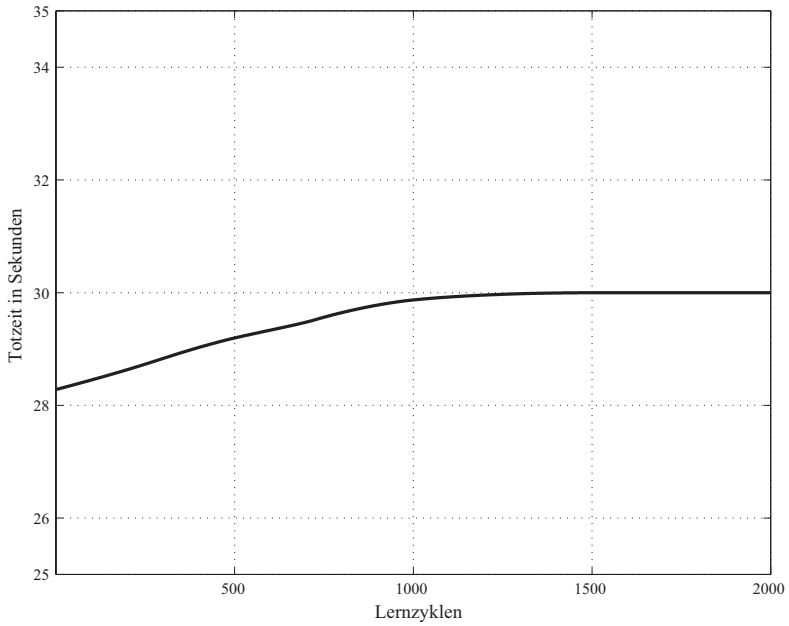

Abbildung 8.8: Identifikationsplot zur Identifikation der Totzeit der Lebensmittelmischanlage. Dargestellt ist die Entwicklung der maximalen Gesamttotzeit des besten EAT-Experten. Die Zufallsinitialisierung der Zeitverzögerungswerte erfolgte gemäß dem vorhandenen Vorwissen über die zu identifizierende Totzeit im Intervall $[\frac{25}{2}; \frac{35}{2}]$.

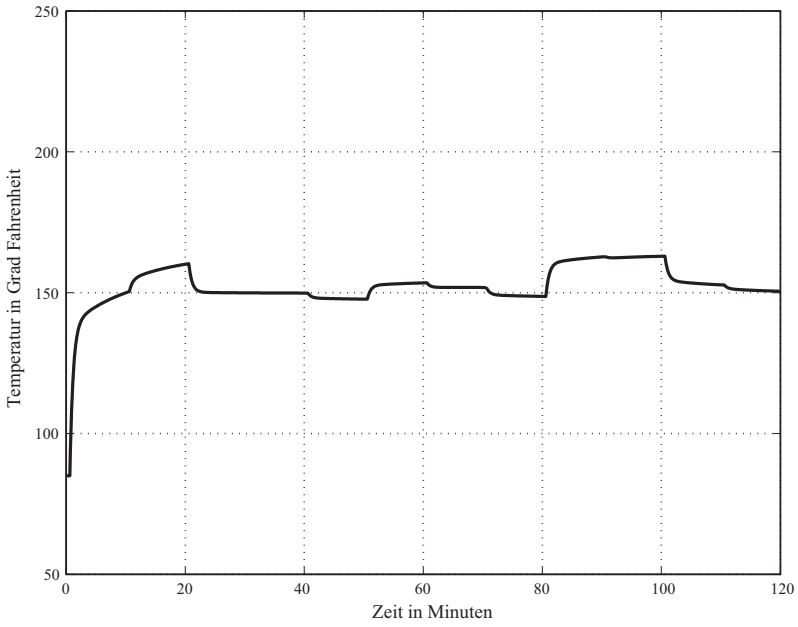

Abbildung 8.9: Verlauf der Temperatur im Mischbehälter bei Verwendung der neuronalen Totzeitidentifikation mit anschließender PINN-Smith-Regelung. Dieser Ansatz ermöglicht einen nahezu stabilen Verlauf der Temperatur im Mischbehälter. Die Kurve steigt von der Umgebungstemperatur von 85 Grad Fahrenheit auf die gewünschte Solltemperatur von 150 Grad Fahrenheit an und behält diese unter geringen Abweichungen bei.

Kapitel 9

Schlußbetrachtung

Dieses abschließende Kapitel faßt die Inhalte der Arbeit zusammen und gibt einen Ausblick auf zukünftige Forschungsprojekte im Umfeld dieser Arbeit.

9.1 Zusammenfassung

Die vorliegende Arbeit zeigt eine neue neuronale Methode zur Identifikation beliebiger Totzeiten in nichtlinearen dynamischen Systemen und zur verbesserten Modellierung und Regelung von Totzeitsystemen. Die Methode basiert auf speziell für diese Zwecke entwickelten neuronalen Netzen, welche als EAT-Netze eingeführt wurden. Die adaptierbaren Parameter der neuronalen Netze werden per Gradientenabstieg anhand von Meßdaten optimiert. Die neuronale Identifikationsmethode erlaubt die Identifikation von Totzeiten allein anhand des Ein-/Ausgabeverhaltens des zugrundeliegenden Totzeitsystems. Die Funktionstüchtigkeit der neu entwickelten neuronalen Methode wurde anhand von drei unterschiedlichen Anwendungsbeispielen demonstriert.

Die Problemstellung der Totzeitidentifikation stammt aus der Regelungstechnik. Totzeiten verursachen, daß die Systemeingabe nicht sofort, sondern erst nach Ablauf einer bestimmten Zeitverzögerung die Systemausgabe beeinflußt. Die Konsequenz ist eine erschwerte Systemregelung. Beim Umgang mit Totzeiten mußte man bisher von vereinfachenden Annahmen ausgehen, beispielsweise von linearen Systemen oder ganzzahligen Totzeiten. Darüberhinaus ist in vielen Anwendungen die genaue Größenordnung der Totzeit nicht bekannt. Mit der in dieser Arbeit entwickelten Methode können *beliebige* Totzeiten in *nichtlinearen dynamischen* Systemen identifiziert werden.

Die Lösung des Problems stammt aus der Neuroinformatik. Neuronale Netze besitzen die Fähigkeit, ihre Parameter anhand von Daten zu optimieren. Diese Fähigkeit wurde bereits

in vielfacher Ausprägung zur Systemmodellierung und -identifikation genutzt. Der Einsatz neuronaler Netze zur Totzeitidentifikation jedoch ist neu. Betrachtet man die Identifikation von Totzeiten als Lernproblem neuronaler Netze, so sind spezielle Netzarchitekturen und Lernverfahren notwendig, die das Auftreten von Totzeiten berücksichtigen können.

In dieser Arbeit wurden die EAT-Netze als neuronale Totzeit-Identifikationsnetze entwickelt. Sie basieren auf den Adaptive Time-Delay Neural Networks (ATNN) [41, 16]. Die EAT-Netze übernehmen im wesentlichen den Aufbau der ATNN. EAT-Netze sind vorwärtsgerichtete neuronale Netze mit internen adaptierbaren Zeitverzögerungsparametern. Im Unterschied zu den ATNN besitzen die EAT-Netze jedoch ein flexibles Lernverfahren, das die Identifikation beliebiger Totzeiten ermöglicht. Dies wird durch die reellwertige Modellierung der Zeitverzögerungsparameter in Kombination mit linearer Interpolation und durch ein neues Aging-Verfahren realisiert. Die EAT-Netze bilden den Kernbaustein der neuronalen Identifikationsmethode.

Den Ausgangspunkt der Methode bildet ein Totzeitsystem mit einer unbekannten Totzeit. Die meßtechnische Erfassung des Totzeitsystems liefert Daten für das Training der EAT-Netze. Nach dem Training steht das EAT-Netz als Modell des Totzeitsystems zur Verfügung. Durch die in dieser Arbeit definierte Interpretation eines EAT-Netzes gelangt man anschließend zur Identifikation der Systemtotzeit. Als Ergebnis liegt ein Totzeitsystem mit der nun bekannten Totzeit vor. Das neu erworbene Wissen kann zur Analyse, Modellierung oder Regelung des Totzeitsystems genutzt werden. Damit trägt die entwickelte neuronale Methode dazu bei, das Verständnis über ein System zu vertiefen und eine optimale Systemregelung zu ermöglichen. Dabei liefert die Identifikationsmethode einfache Bewertungskriterien, die es dem Anwender erlauben, die Zuverlässigkeit des gelieferten Identifikationsergebnisses zu beurteilen und daraufhin selbst zu entscheiden, ob das Ergebnis als vertrauenswürdig eingestuft werden kann. Die Möglichkeit, die Konfidenz der neuronalen Methode bewerten zu können, ist eine wichtige Voraussetzung für deren praktischen Einsatz.

Bei vielen Anwendungen ist die Systemtotzeit nicht gänzlich unbekannt. Meist liegt Vorwissen über die zu identifizierende Totzeit in Form eines möglichen Wertebereichs vor. Die neuronale Identifikationsmethode ermöglicht durch eine gezielte Initialisierung der Zeitverzögerungsparameter der EAT-Netze ein einfach realisierbares und sehr effektives Verfahren zur Nutzung von Vorwissen. Das Ergebnis ist nicht nur eine exaktere Identifikationsleistung, sondern auch signifikant verkürzte Trainingszeiten.

Bei der meßtechnischen Erfassung eines Systems können zahlreiche Störungen auftreten. Es können hardwarebedingte Ausreißer oder Signalverluste vorkommen oder die Meßeinrichtungen an sich verursachen Rauschen. Verrauschte Daten jedoch stellen die neuronale Identifikationsmethode vor prinzipielle Schwierigkeiten. Neuronale Netze sind datengetrieben, d.h. sie versuchen anhand der Messungen auf den zugrundeliegenden Datengenerierungsprozeß zu schließen. Verrauschte Daten stellen aber ein *verfälschtes* Abbild des zugrundliegenden Systems dar, so daß das neuronale Netz zusätzlich zur eigentlichen Identifikations-

leistung das Rauschen filtern muß, um dennoch die korrekte Totzeit zu identifizieren. Die in dieser Arbeit entwickelte Methode hat sich in den dargestellten Versuchen als robust gegenüber Rauschen erwiesen.

Ein weiterer Aspekt bei der Datenerfassung ist die Wahl der Abtastrate. Die Abtastrate bestimmt, in welchem zeitlichen Abstand die einzelnen Datenpunkte aufeinander folgen. Innerhalb eines gegebenen Meßzeitraums bestimmt die Abtastrate damit die Anzahl der zur Verfügung stehenden Datenpunkte. Untersuchungen zu unterschiedlichen Abtastraten haben ergeben, daß ein zu grobes Datenraster das Identifikationsergebnis signifikant verschlechtert, da zu wenig Datenpunkte zur Identifikation vorliegen. Das Problem tritt aber erst nach mehrmals iterierter Datenausdünnung auf. In der Regel reichen die zur Verfügung stehenden Daten für die Totzeitidentifikation aus. Bei schlechten Identifikationsergebnissen jedoch lohnt sich ein prüfender Blick auf die Wahl der Abtastrate.

Eine Totzeit kann zeitvariabel sein, d.h. ihr Wert kann sich während des Betriebs ändern. Im Falle zeitvarianter Totzeiten liefert die neuronale Identifikationsmethode den Wertebereich, innerhalb dessen sich die Totzeit bewegt. Dies stellt einen wertvollen Beitrag zum besseren Verständnis über das Totzeitsystem dar.

Die neuronale Identifikationsmethode liefert auch dann ein korrektes Ergebnis, wenn Daten vorliegen, die keine Totzeit beinhalten. Während der zu dieser Arbeit durchgeführten Versuche stellte sich das neuronale Identifikationsverfahren sogar als prüfende Instanz heraus. Anhand gegebener Daten, die eine Totzeit beinhalten sollten, konnte das Verfahren keine Totzeit identifizieren. Nach genauerer Untersuchung des Datengenerierungsprozesses stellte sich heraus, daß die Daten tatsächlich keine Totzeit aufweisen konnten.

Schließlich liefert die neuronale Identifikationsmethode auf halbem Weg ein anderweitig sinnvoll verwendbares Zwischenprodukt. Die EAT-Netze werden anhand von Meßdaten trainiert. Nach dem Training steht das EAT-Netz als Modell des Totzeitsystems zur Verfügung. Dieses neuronale Modell kann innerhalb eines intelligenten Reglers eingesetzt werden. Eine entsprechende Anwendung aus der Automobilindustrie verdeutlichte, wie mit einem derartigen Ansatz eine verbesserte Regelung erreicht werden kann.

Insgesamt betrachtet steht mit der in dieser Arbeit entwickelten neuronalen Methode ein Werkzeug zur Verfügung, das dazu beiträgt, das Verständnis über ein Totzeitsystem zu vertiefen und damit eine verbesserte Systemmodellierung und -regelung zu ermöglichen.

9.2 Ausblick

Systeme mit Totzeiten treten in den unterschiedlichsten Anwendungen auf. Es gibt kaum Bereiche, in denen Totzeitphänomene a priori ausgeschlossen werden können. Die am meisten zitierten Bereiche sind Verfahrenstechnik, Biologie, Medizin, Ökologie und Ökonomie.

Diese Arbeit zeigte Anwendungsbeispiele aus der Mikroelektronik, der Automobilindustrie und der Lebensmitteltechnologie. Die in dieser Arbeit vorgestellte neuronale Methode zur Identifikation von Totzeiten und zur Modellierung und Regelung von Totzeitsystemen ist nicht beschränkt auf spezifische Einzelanwendungen. Sie ist generell anwendbar auf nichtlineare dynamische totzeitbehaftete Systeme. Anwendungspotential ergibt sich grundsätzlich bei komplexen, hochdynamischen Prozessen, über die geringes oder gar kein Vorwissen vorliegt. Einzige Voraussetzung zur Anwendbarkeit der neuronalen Methode ist die meßtechnische Erfaßbarkeit des Totzeitsystems. Im Folgenden wird ein Ausblick auf mögliche, an diese Arbeit anknüpfende Ansätze zu zukünftigen Forschungsarbeiten gegeben.

Mit der in dieser Arbeit entwickelten neuronalen Methode steht ein eigenständiges Werkzeug zur Identifikation von Totzeiten anhand des Ein-/Ausgabeverhaltens des betrachteten Totzeitsystems zur Verfügung. Dennoch könnten sich Untersuchungen lohnen, dieses Werkzeug in allgemeinere Ansätze zu integrieren. In der Dissertation [8] beispielsweise stellt T. Brychcy mit den sogenannten Systemtheoretisch Vorstrukturierten Neuronalen Netzen (SVNN) eine sehr allgemeine Methode zur Identifikation unbekannter Nichtlinearitäten in dynamischen Systemen vor. Diese SVNN beruhen auf der ebenfalls in dieser Arbeit definierten Klasse der Modularen Allgemeinen Rekurrenten Neuronalen Netzen (MARNN). Die MARNN ermöglichen die strukturerhaltende Abbildung eines Blockdiagramms auf ein rekurrentes neuronales Netz. Die Neuronen der MARNN können komplexe Subnetze mit optimierbaren Parametern sein. Liegt also ein System als Blockdiagramm vor, so erlaubt die Methode von Brychcy die Identifikation unbekannter, nichtlinearer Teilsysteme dieses (Gesamt-)Systems. In der Dissertation [37] werden die MARNN um den neuen Neuronentyp der Preisach-Neuronen erweitert. Die Preisach-Neuronen ermöglichen die Identifikation von Systemen mit Hysterese.

Systemtheoretisch Vorstrukturierte Neuronale Netze sind eine zeitdiskrete Approximation des in Form eines Blockdiagramms gegebenen zeitkontinuierlichen Systems. Die Behandlung von Totzeiten in MARNN beschränkt sich auf Totzeiten, die ein ganzzahliges Vielfaches der zeitlichen Diskretisierung darstellen. Ein Einbau der EAT-Netze als neuer Neuronentyp der MARNN würde die Identifikation beliebiger Totzeiten in SVNN ermöglichen. Allerdings verfügen die EAT-Netze über einen Zustand variabler Länge. Eine Anforderung an Subnetze in MARNN ist aber ein Zustand fester Länge. Es wäre zu untersuchen, ob der Einbau in MARNN die Mächtigkeit der EAT-Netze einschränkt oder ob MARNN mit EAT-Neuronen ohne Performanzverlust in der Lage sind, beliebige Totzeiten zu identifizieren. Weiterhin könnte untersucht werden, ob beispielsweise die gleichzeitige Identifikation von Hysterese und Totzeiten möglich oder ob deren sequentielle Identifikation sinnvoller ist.

Bei der Systemidentifikation wird unterschieden, ob diese on-line oder off-line durchgeführt wird. Die in dieser Arbeit entwickelte neuronale Methode wird off-line durchgeführt, d.h. es werden Meßdaten vom System gesammelt und diese später zum Training der EAT-Netze verwendet. Eine sinnvolle Erweiterung der Methode wäre deren Einsatz für den On-line-

Betrieb. On-line bedeutet, daß die Identifikation der Totzeit im laufenden Betrieb abläuft. Da das Lernverfahren der EAT-Netze Einzelschrittverarbeitung realisiert, ist es prinzipiell möglich, diese Netze on-line zu belernen. Allerdings ist es fraglich, ob das Erlernen der Totzeiten schnell genug für einen On-Line-Betrieb ist. Unter Umständen müssen konvergenzbeschleunigende Maßnahmen, wie etwa die Berücksichtigung höherer Ableitungen, entwickelt werden.

In der Einleitung dieser Arbeit wurde dargelegt, daß Modelle technischer Systeme immer größeren Ansprüchen genügen müssen. Dies zieht unweigerlich eine immer genauere Identifikation der Systeme nach sich. Diese Arbeit leistet nicht nur einen Beitrag zur Lösung der Totzeitproblematik, sondern liefert auch eine wichtige Basis für weitere Fortschritte auf den Gebieten der Systemanalyse, -modellierung und -regelung.

Literatur

— **Anmerkung der Autorin:** Aufgrund meiner Eheschließung hat sich mein Nachname während des Promotionsverfahrens geändert. Meine Veröffentlichungen sind daher unter *Ungerer (verh. Brand)* zu finden.

[1] Anton, M.: *Systematischer Einsatz von Approximations- und Syntheseverfahren für die Modellbildung in der Mikrosystemtechnik.* Dissertation, Institut für Theoretische Elektrotechnik und Mikroelektronik, Universität Bremen 1999.

[2] Ayoubi, M.: Das dynamische Perzeptronmodell zur experimentellen Modellbildung nichtlinearer Prozesse. In *Informatik Forschung und Entwicklung*, Bd. 12. Springer, 1997, S. 14–22.

[3] Ayoubi, M. und Isermann, R.: Radial basis function networks with distributed dynamics for nonlinear dynamic system identification. In *Proceedings of the 3rd European Congress on Intelligent Techniques and Soft Computing, EUFIT'95*, Aachen, 1995.

[4] Böttiger, A.: *Regelungstechnik - Eine Einführung für Ingenieure und Naturwissenschaftler.* 3. Auflage. München: Oldenbourg 1998.

[5] Brauer, W., Brychcy, T., Kirchmair, C., Sturm, M., und Ungerer (verh. Brand), C.: Abschlußbericht zum Projekt ACON - Adaptive Control. *Forschungsberichte Künstliche Intelligenz: FKI-239-00*, Hrsg. Institut für Informatik, Technische Universität München 2000.

[6] Bronstein, I. N. und Semendjajew, K. A.: *Taschenbuch der Mathematik.* 25. Auflage. Thun: Harri Deutsch 1991.

[7] Brown, M. und Harris, C.: *Neurofuzzy Adaptive Modelling and Control.* Systems and Control Engineering. London: Prentice Hall 1994.

[8] Brychcy, T.: *Modellierung dynamischer Systeme mit vorstrukturierten neuronalen Netzen.* Dissertation, Institut für Informatik, Lehrstuhl Prof. Dr. Dr. h.c. W. Brauer, Technische Universität München 2000. (Infix Verlag).

[9] Busch, P.: *Elementare Regelungstechnik.* Würzburg: Vogel Buchverlag 1995.

[10] Butz, D.: *Neuronale Funktionsapproximation mit RBF-Schwerpunktnetzen.* Dissertation, Institut für Informatik, Lehrstuhl Prof. Dr. Dr. h.c. W. Brauer, Technische Universität München 1997. (Shaker Verlag).

[11] Camacho, E. F. und Bordons, C.: *Model Predictive Control in the Process Industry.* London: Springer 1999.

[12] Christen, U.: *Engineering Aspects of H_∞ Control.* Dissertation, ETH Zürich 1996.

[13] Cichocki, A. und Unbehauen, R.: *Neural Networks for Optimization and Signal Processing.* Stuttgart: Teubner 1993.

[14] Cremer, M.: *Regelungstechnik - Eine Einführung*, Bd. 2. Berlin: Springer 1995.

[15] Day, S. P. und Camporese, D. S.: Continuous-time temporal backpropagation. In *Int. Joint Conf. Neural Networks*, Bd. 2, Seattle, WA, 1991, S. 95–100.

[16] Day, S. P. und Davenport, M. R.: Continous-time temporal back-propagation with adaptable time delays. *IEEE Transactions on Neural Networks*, *Bd. 4 [1993]* Nr. 2, S. 348–354.

[17] Doyle, J. C., Glover, K., Khargonekar, P., und Francis, B. A.: State space solutions to standard H_2 and H_∞ control problems. *IEEE Transactions on Automatic Control*, *Bd. AC-34* [1989], S. 831–847.

[18] Fagerer, C.: *Automatische Teleoperation eines Tracking- und Greifvorgangs im Weltraum basierend auf Bilddatenauswertung.* Dissertation, Universität der Bundeswehr München 1996.

[19] Francis, B. A.: *A Course in H_∞ Control Theory.* Berlin: Springer 1987.

[20] Frank, P. M.: *Regler mit negativer Gruppenlaufzeit für Totzeitstrecken.* Dissertation, Technische Hochschule Karlsruhe 1966.

[21] Garcia, C. E., Prett, D. M., und Morari, M.: Model predictive control: Theory and practice – A survey. *Automatica*, *Bd. 25 [1989]* Nr. 3, S. 335–348.

[22] Gorecki, H., Fuksa, S., Grabowski, P., und Korytowski, A.: *Analysis and Synthesis of Time Delay Systems.* Chichester: John Wiley & Sons 1989.

[23] Habetler, T.: A space vector-based rectifier regulator for ac/dc/ac converters. *IEEE Transactions on Power Electronics*, *Bd. Vol. 8, No. 1* [1993], S. 30–36.

[24] Hale, J. K. und Lunel, S. M.: *Introduction to Functional Differential Equations.* Springer 1993.

[25] Haykin, S.: *Neural Networks*, Bd. 1. New York: Macmillan 1994.

[26] Hebb, D.: *The Organization of Behaviour.* New York: Wiley 1949.

[27] Hänsler, E.: *Statistische Signale.* 2. Auflage. Berlin: Springer 1997.

[28] Hoffmann, J.: *MATLAB und SIMULINK - Beispielorientierte Einführung in die Simulation dynamischer Systeme.* Bonn: Addison-Wesley-Longman 1998.

[29] Hornik, K., Stinchcombe, M., und White, H.: Multilayer feedforward networks are universal approximators. *Neural Networks*, *Bd. 2* [1989], S. 359–366.

[30] Horáček, P.: A laboratory platform for continuing control education. In *Proceedings of the IFAC/IEEE Symposium on Advances in Control Education ACE2000*, Gold Coast, Australia, 2000.

[31] Hunt, K. J., Sbarbaro, D., Żbikowski, R., und Gawthrop, P. J.: Neural networks for control systems – A survey. *Automatica*, *Bd. 28 [1992]* Nr. 6, S. 1083–1112.

[32] Jankovic, M. und Kolmanovsky, I.: Controlling nonlinear systems through time delays: An automotive perspective. In *Proceedings of the European Control Conference, ECC'99*, Karlsruhe, 1999.

[33] Jenni, F. und Wüest, D.: *Steuerverfahren für selbstgeführte Stromrichter.* vdf Hochschulverlag ETH Zürich und Teubner Stuttgart 1995.

[34] Jörgl, H. P.: *Repetitorium Regelungstechnik*, Bd. 1. Wien, München: Oldenburg 1993.

[35] Jörgl, H. P.: *Repetitorium Regelungstechnik*, Bd. 2. Wien, München: Oldenburg 1993.

[36] Karg, E.: *Regelungstechnik.* 7. Auflage. Kamprath-Reihe. Würzburg: Vogel 1992.

[37] Kirchmair, C.: *Identifikation von Systemen mit Hysterese mit Hilfe von Preisach-Neuronen in vorstrukturierten neuronalen Netzen.* Dissertation, Institut für Informatik, Lehrstuhl Prof. Dr. Dr. h.c. W. Brauer, Technische Universität München 2001. (Infix Verlag).

[38] Kolmanovskii, V. und Myshkis, A.: *Applied Theory of Functional Differential Equations.* Dordrecht: Kluwer Academic Publishers 1992.

[39] Lang, K. J. und Hinton, G. E.: A time-delay neural network architure for speech recognition. *Technical Report: CMU-CS-88-152*, Hrsg. Carnegie-Mellon University 1988.

[40] Leonhard, W.: *Einführung in die Regelungstechnik.* Braunschweig: Vieweg 1992.

[41] Lin, D.-T., Dayhoff, J., und Ligomenides, P.: A learning algorithm for adaptive time-delays in a temporal neural network. *Technical Report: TR92-59*, Hrsg. University of Maryland 1992.

[42] Lin, D.-T., Dayhoff, J., und Ligomenides, P.: Learning spatiotemporal topology using an adaptive time-delay neural network. In *Proceedings of the World Congress on Neural Networks*, Bd. 1, Portland, OR, 1993, S. 291–294.

[43] Lin, D.-T., Dayhoff, J., und Ligomenides, P.: Trajectory production with the adaptive time-delay neural network. *Neural Networks*, *Bd. 8 [1993]* Nr. 3, S. 447–461.

[44] Malek-Zavarei, M. und Jamshidi, M.: *Time Delay Systems: Analysis, Optimization and Applications*. Amsterdam: North-Holland 1987.

[45] McCulloch, W. und Pitts, W.: A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *Bd. 5* [1943], S. 115–133.

[46] Miller, W. T., Sutton, R. S., und Werbos, P. J. (Hrsg.): *Neural Networks for Control*. Cambridge, Mass.: MIT Press 1990.

[47] Mills, P. M., Zomaya, A. Y., und Tadé, M. O.: *Neuro-Adaptive Process Control - A Practical Approach*. Chichester: Wiley 1996.

[48] Moody, J. und Darken, C. J.: Fast learning in networks of locally-tuned processing units. *Neural Computation*, *Bd. 1* [1989], S. 281–294.

[49] Narendra, K. S. und Parthasarathy, K.: Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, *Bd. 1 [1990]* Nr. 1, S. 4–27.

[50] Parks, T. W. und Burrus, C. S.: *Digital Filter Design.* 1. Auflage. New York: John Wiley & Sons. Inc. 1987.

[51] Pleßmann, K.-W. 2000. Systemanalyse für die Automatisierung komplexer Systeme. VDI Bildungswerk. Handbuch zum VDI-Seminar 38-74-03.

[52] Poggio, T. und Giorosi, F.: A theory of networks for approximation and learning. *A.I. Memo: No. 1140*, Hrsg. Massachusetts Institute of Technology, Artificial Intelligence Laboratory 1989.

[53] Powell, M. J. D.: Radial basis functions for multivariable interpolation: A review. In *Proc. IMA Conf. on "Algorithms for the approximation of functions and data"*. RMCS, Shrivenham, 1985.

[54] Riedmiller, M.: Rprop - description and implementation details. *Technischer Bericht*, Hrsg. Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe 1994.

[55] Riedmiller, M. und Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proceedings of the IEEE International Conference on Neural Networks 1993 (ICNN 93)*, 1993.

[56] Roduner, C. A.: *H_∞-Regelung linearer Systeme mit Totzeiten.* Dissertation, ETH Zürich 1998.

[57] Rojas, R.: *Neural networks: a systematic introduction.* Berlin: Springer 1996.

[58] Rosenblatt, F.: The perceptron: A perceiving and recognizing automaton. *Report: 85-460-1*, Hrsg. Project PARA, Cornell Aeronautical Laboratory, Ithaca, New York 1957.

[59] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *Bd. 65* [1958], S. 386–408.

[60] Rugh, W. J. und Shamma, J. S.: Research on gain scheduling. *Automatica*, *Bd. 36* [2000], S. 1401–1425.

[61] Rumelhart, D. E. und McClelland, J. L.: *Parallel Distributed Processing*, Bd. 1. MIT Press 1986.

[62] Saad, D. (Hrsg.): *On-line Learning in Neural Networks.* Publications of the Newton Institute. Cambridge: Cambridge University Press 1998.

[63] Samal, E. und Becker, W.: *Grundriss der praktischen Regelungstechnik.* 20. Auflage. München: Oldenbourg 2000.

[64] Schmidt, R. F., Dudel, J., Jänig, W., und Zimmermann, M.: *Grundriß der Neurophysiologie.* 6. Auflage. Berlin: Springer 1987.

[65] Schoen, G. M.: *Stability and Stabilization of Time-delay Systems.* Dissertation, ETH Zürich 1995.

[66] Schröder, D.: *Elektrische Antriebe 4, Leistungselektronische Schaltungen.* Berlin: Springer 1998.

[67] Schultz, J.: *Identifikation nichtlinearer dynamischer Systeme mit künstlichen neuronalen Netzen.* Dissertation, Universiät Karlsruhe 1998.

[68] Seborg, D. E.: A perspective on advanced strategies for process control. In Frank, P. M. (Hrsg.), *Advances in Control.* Springer, Berlin, 1999, S. 103–134.

[69] Smith, O.: A controller to overcome dead time. *ISA Journal*, *Bd. 6* [1959].

[70] Sturm, M.: Simulation einer Gebäudeheizung. *Forschungsberichte Künstliche Intelligenz: FKI-227-98*, Hrsg. Lehrstuhl Prof. Dr. Dr. h.c. W. Brauer, Institut für Informatik, Technische Universität München 1998.

[71] Sturm, M.: *Neuronale Netze zur Modellbildung in der Regelungstechnik.* Dissertation, Institut für Informatik, Lehrstuhl Prof. Dr. Dr. h.c. W. Brauer, Technische Universität München 2000. (Eigenverlag, On-Line: `http://tumb1.biblio.tu-muenchen.de/publ/diss/in/2000/sturm.pdf`.).

[72] Sturm, M., Brychcy, T., und Kirchmair, C.: AMoC - The ACON Model Classes. *Forschungsberichte Künstliche Intelligenz: FKI-224-97*, Hrsg. Institut für Informatik, Technische Universität München 1997.

[73] The MathWorks 1995. *SIMULINK Dynamic System Simulation Software — User's Guide.* 4. Auflage.

[74] The MathWorks 1996. *MATLAB High-Performance Numeric Computation and Visualization Software — User´s Guide.* 4. Auflage.

[75] Ungerer (verh. Brand), C.: Neuronale Modellierung von Totzeiten in nichtlinearen dynamischen Systemen. Diplomarbeit, Institut für Informatik, Lehrstuhl Prof. Dr. Dr. h.c. W. Brauer, Technische Universität München 1998.

[76] Ungerer (verh. Brand), C.: Identifying time-delays in nonlinear control systems: An application of the adaptive time-delay neural network. In Mohammadian, M. (Hrsg.), *CIMCA ´99 - Neural Networks & Andvanced Control Strategies.* IOS Press, Amsterdam, 1999, S. 99–104.

[77] Ungerer (verh. Brand), C.: Motor control using adaptive time-delay learning. In Kurková, V. (Hrsg.), *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms.* Springer, Wien, 2001, S. 332–335.

[78] Ungerer (verh. Brand), C., Kirchmair, C., Stübener, D., und Sturm, M.: Supporting traditional controllers of combustion engines by means of neural networks. In Reusch, B. (Hrsg.), *Computational Intelligence - Theory and Applications, Proceedings of the 6th Fuzzy Days.* Springer, Berlin, 1999, S. 132–141.

[79] Ungerer (verh. Brand), C. und Radan, A.: Identifying time delays using neural networks. In Tsaptsinos, D. (Hrsg.), *Engineering Problems - Neural Network Solutions, Proceedings of the International Conference on Engineering Applications of Neural Networks (EANN 2000)*, Kingston, 2000, S. 237–244.

[80] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., und Lang, K.: Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *Bd. 37* [1989], S. 328–339.

[81] Wan, E. A.: Temporal backpropagation for FIR neural networks. In *International Joint Conference on Neural Networks*, San Diego, CA, 1990, S. 575–580.

[82] Wan, E. A.: Time series prediction by using a connectionist network with internal delay lines. In Weigend, A. und Gershenfeld, N. (Hrsg.), *Time Series Prediction. Forecasting the Future and Understanding the Past*, Bd. XVII. Addison-Wesley, SFI Studies in the Sciences of Complexity, 1994.

[83] Werbos, P. J.: Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, *Bd. 78 [1990]* Nr. 10, S. 1550–1560.

[84] Weweris, R.: *Aktive Beeinflussung selbstregelnder Schwingungen bei Systemen mit Totzeit*. Dissertation, Universität - Gesamthochschule - Duisburg 1994.

[85] White, D. A. und Sofge, D. A. (Hrsg.): *Handbook of Intelligent Control*. New York: Van Nostrand Reinhold 1992.

[86] Widrow, B. und Winter, R.: Neural nets for adaptive filtering and adaptive pattern recognition. *Computer (IEEE)*, *Bd. 21 [1988]* Nr. 3, S. 25–39.

[87] Williams, R. J. und Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. *Technischer Bericht: ICS Report 8805*, Hrsg. Institute for Cognitive Science, University of California, San Diego, CA 1988.

[88] Zell, A.: *Simulation neuronaler Netze*. Bonn: Addison-Wesley 1994.

[89] Zirpel, M.: *Regelungstechnik mit PC-Simulation*. Poing: Franzis 1994.